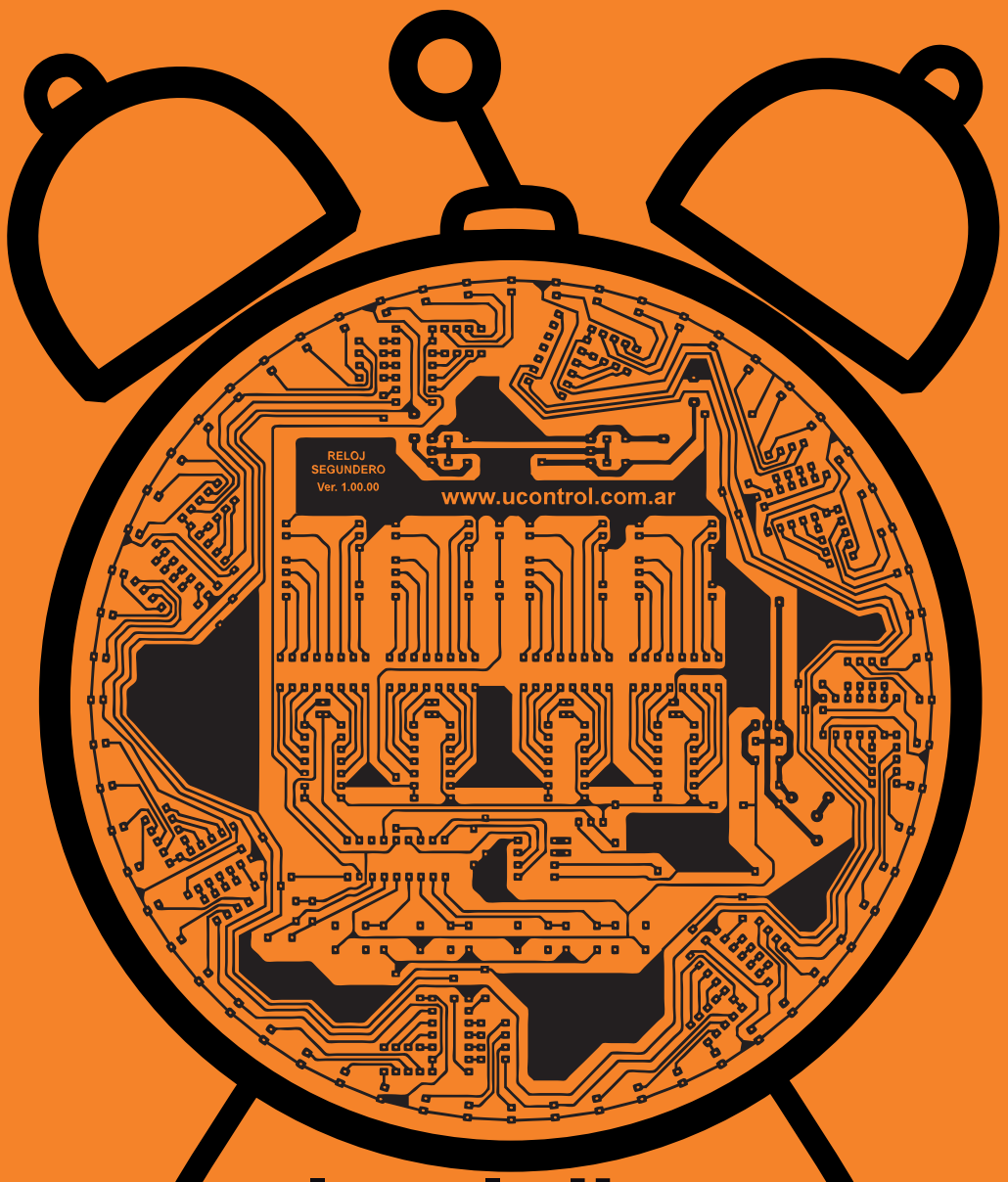
// DESCARGA GRATUITA //

“int año = 1, número 2;” /* MARZO 2008 */

µCONTROL

Electrónica en General Pics en Particular



El Relojito II

Analizaremos las rutinas necesarias para la programación de nuestro reloj. Lenguajes PIC BASIC y CCS

El u-Scilador

Capaz de trabajar con frecuencias comprendidas entre uno y varias decenas de MHz

Estreno!!!

Te presentamos las nuevas secciones: Paleotrónica y Ludoteca

.indice

llave accionada al tacto	0x04
el Protocolo Wiegand	0x05
PICs y LEDs: una pareja perfecta	0x08
poniendo un poco de orden en C	0x0B
simulador de circuitos digitales	0x0D
el relojito. segunda parte	0x10
PIC BASIC. CAP. II	0x13
hablemos de antenas	0x15
receptor para el protocolo DMX512	0x19
monoestable con NE555	0x1F
el bus SPI	0x21
el u-Scilador	0x26
dado electrónico. versión 1	0x28
dado electrónico con PIC 16	0x2B
el microprocesador INTEL 4004	0x2E

.staff

hicimos este número...

número = 2; año = 1;

Dirección y Redacción:
Ariel Palazzesi
Argentina
arielpalazzesi@gmail.com
www.ucontrol.com.ar

Edición, Redacción y Corrección:
Reinier Torres Labrada
Cuba
reiniertl@gmail.com

Diseño:
Verónica C. Lavore
Argentina
azimut.estudio@gmail.com

Consejo Editorial:
Mario Sacco
Argentina
service.servisystem@gmail.com

Carlos Ortega Sabio
España
carlos.ortegasabio@ucontrol.com.ar

Diego Márquez García - Cuervo
picmania@garcia-cuervo.com
http://picmania.garcia-cuervo.net/

Marcos Lazcano
Argentina
marcos.lazcano@gmail.com

Pedro
Venezuela
palitroquez@gmail.com

Contacto:
revista@ucontrol.com.ar
www.ucontrol.com.ar

.editorial

Nuevamente estoy sentado frente al ordenador, enfrentado a la tarea de escribir la editorial de la revista. Lo primero que se me ocurre es que, en esta oportunidad, más que “Editorial” deberíamos poner a este espacio el nombre de “Agradecimientos”. Veamos porque.

En primer, es imprescindible agradecer a los más de 14.000 lectores que descargaron la revista en los primeros dos meses. No hace falta decir que un número semejante ha superado con creces nuestras más locas fantasías. El sorteo del GTP-USB+ contó con mas de 700 participantes, y nuestra casilla de correo se vio inundada de mensajes expresando felicitaciones, agradecimientos y sugerencias (que serán debidamente tomadas en cuenta) para los próximos números.

En segundo lugar tengo que reconocer la calidad y el esfuerzo de los colaboradores que han hecho posibles estos dos números de uControl. Probablemente no lo sepas, pero quienes escriben en uControl lo hacen restando tiempo a sus labores habituales, sin cobrar un centavo por ello. Este hecho, sin duda, hace aún más valiosos sus artículos.

Algo que también nos ha sorprendido gratamente, y que necesario agradecer, es la buena aceptación que ha tenido esta modestísima revista en los foros y blogs de todo el mundo. Muchos sitios han colocado una copia del documento PDF para que sus lectores lo puedan descargar directamente desde sus servidores, y algunos incluso nos han honrado abriendo una sección dedicada a esta publicación.

Todo esto hace que el compromiso asumido en el número anterior, de entregar cada 64 días una revista prolija, fresca y útil, se vea fortalecido. Haremos todo lo posible por que cada número sea mejor que el anterior, sumando cantidad de artículos y de páginas para que todos encuentren en ella el dato que buscan. Para lograrlo, hemos hecho de este segundo número es una evolución del primero. Hay nuevas secciones, hemos intentando mejorar la forma en que escribimos para que los textos sean fácilmente comprensibles, y el diseño de las páginas e imágenes también ha sido revisado para hacer mas agradable la lectura.

Este mes estrenamos colaboradores nuevos. En realidad, por motivos de tiempo nos han quedado sin incluir varios artículos de gran calidad y valor didáctico que colaboradores espontáneos nos han enviado. Por supuesto, podrás leerlos en los números siguientes o en nuestro sitio Web.

En uControl esperamos que la revista que acabas de descargar sea de tu agrado. Si al menos una pagina te resulta de utilidad, el esfuerzo habrá valido la pena.

¡Hasta el próximo Número! ■

//Mario Sacco//
service.servisystem@gmail.com

llave accionada al tacto

Es posible reemplazar nuestros viejos pulsadores utilizando solo un puñado de componentes fáciles de conseguir. Utilizaremos botones metálicos, eternos, que darán una nueva dimensión a nuestras posibilidades de desarrollo. El circuito que presentamos aquí puede utilizarse en casi todos los proyectos que requieran el accionar de interruptores.

Al levantarnos, por la mañana, apagamos el despertador (pulsamos un botón), encendemos la luz (pulsamos un botón), vamos al baño y encendemos la luz (pulsamos un botón), nos aseamos y al salir apagamos la luz (pulsamos un botón).

Aún no sabemos si ha salido el sol o está nublado y ya hemos pulsado botones una innumerable cantidad de veces.

Cuando uno de estos botones que acompañan cada día de nuestras vidas comienza a fallar, comenzamos a maltratarlo hasta que logramos que el artefacto en cuestión nos haga el honor de ejecutar la función que deseamos. Este es un hecho estadística comprobable ¿a quién no le ha pasado alguna vez?

Cuando dicho aparato ya no puede soportar más golpes, terminamos dándonos cuenta de la importancia que tiene un sistema como el que describiremos en este artículo. Se trata de un sistema totalmente estanco, sin partes mecánicas o móviles, que será eterno.

Nuestro circuito necesita de un elemento metálico, conductor, que haga las veces de sensor de toque. Hemos utilizado para ello la cabeza de un tornillo, aunque es posible utilizar alguna otra cosa en su lugar. Al tocar este elemento estamos induciendo en el mismo una señal de 50 Hz. que será rectificada por los diodos identificados como D1 y D2 en el esquema. Esto provocara que un voltaje negativo se haga presenten en condensador C3, el resistor R3 y la compuerta (ó gate) del transistor Q1. Esto hace que el transistor pase a un estado de corte.

Nuestro circuito necesita de un elemento metálico, conductor, que haga las veces de sensor de toque.

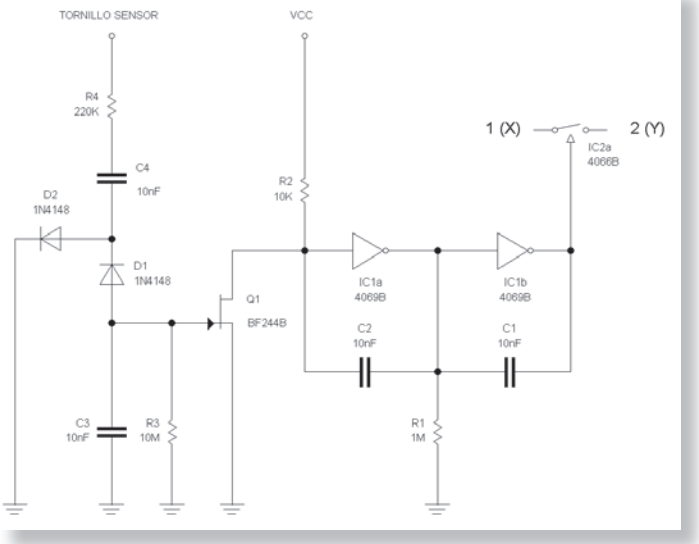
A través del resistor R2 proporcionamos un estado lógico alto a la entrada de la primer puerta del CD4069.

Este “1” es invertido por la segunda compuerta y aplicado a la entrada de control de una de las cuatro llaves analógicas que posee el circuito integrado CD4066. Estas entradas requieren de un estado alto para activar la llave analógica interna.

Dado que un CD4069 dispone de seis compuertas en su interior, y que un CD4066 tiene cuatro llaves, podemos construir tres pulsadores de este tipo con solo dos circuitos integrados. Es importante recordar que con estas llaves podremos conmutar señales de baja tensión y corriente. Esto incluye señales de audio, video, circuitos de continua, señales TTL o CMOS y toda aquella aplicación que no requiera consumos de más de 60 mA. Si el aparato a conmutar consuma una corriente mayor, dañaremos la llave del CD4066.

La versatilidad de este circuito nos permitirá diseñar módulos selectores de funciones, controles de volumen, pulsadores de reset, teclados, etc. Todo aquel proyecto que requiera el uso de un botón pulsador, dejará de tener un elemento que algún día pueda fallar.

En la próxima entrega veremos cómo, con un pequeño agregado, podemos transformar este “pulsador momentáneo”, en un “pulsador con retención”, pudiendo sustituir también a esta clase de frágiles dispositivos.



Es notable que tan pocos componentes, soluciones problemas eternos.

//Diego Márquez García - Cuervo//
picmania@garcia-cuervo.com

el protocolo Wiegand

Una visión general de qué es, para qué sirve y cómo se utiliza el protocolo que implementaron los lectores de tarjetas de Efecto Wiegand.

Primero y antes de empezar, es importante no confundir el Protocolo Wiegand con el Efecto Wiegand. El Efecto Wiegand es un concepto físico en el que intervienen las distintas formas en que reaccionan magnéticamente distintas áreas de un hilo conductor ante la influencia de un campo magnético.

En base a este Efecto Wiegand se construyeron ciertos tipos de tarjetas de identificación y sus correspondientes lectores de tarjetas de proximidad, para usarlos en control de accesos y/o presencia. Estos lectores de tarjetas debían conectarse a los dispositivos de control de acceso de algún modo (véase "Fundamentos de la Comunicación *Síncrona*" en el número anterior), y en lugar de usar algún protocolo ya existente se decidió desarrollar uno propio. Así nació el Protocolo Wiegand, que es al que se refiere este artículo.

Como todo protocolo de comunicaciones el Wiegand consta de dos partes fundamentales. Por un lado tenemos una descripción del modo en que físicamente se transmite la información digital, y por otro la forma de interpretar numéricamente dicha información.

Probablemente debido a mis propias limitaciones y/o desconocimiento no sería capaz de deciros si el protocolo Wiegand es un protocolo serie Síncrono o Asíncrono ya que es fundamentalmente distinto a los que conozco de estos dos tipos anteriores: El ABA Track II como ejemplo de Síncrono o el RS-232 del Asíncrono. En cualquier caso, describiré como funciona y ustedes mismos podrán decidir como llamarlo.

. Sistema de transmisión

La transmisión de datos Wiegand usa tres hilos. La línea para enviar los unos lógicos o DATA1, la línea para hacer lo propio con los ceros lógicos o DATA0 y la línea de masa de referencia de ambos o GND. Los niveles que se usan son Bajo, a nivel de GND, o Alto a +5V o VCC.

En estado de reposo, o sea, sin transmitir, la línea de GND es exactamente lo que es: GND, y siempre está en bajo, por lo que ya no nos referiremos más a ella. Las líneas DATA1 y DATA0 están en alto, a nivel de +5V ó VCC.

Para transmitir un Bit 1 lo que se hace es mandar un pulso a Bajo, normalmente de 50 uS (microsegundos) de duración, por la línea DATA1, mientras DATA0 permanece en Alto.

Por el contrario, para transmitir un Bit 0 lo que se hace es mandar un pulso Bajo, también de la misma duración 50 uS (microsegundos), por la línea DATA0, mientras ahora es DATA1 la que permanece en Alto.

Normalmente la separación entre cada pulso y el siguiente es de unos 2 mS (milisegundos).

Como podéis ver, y a diferencia de los protocolos mencionados anteriormente, los dos tipos de Bits, ceros y unos, son transmitidos de forma idéntica aunque por líneas distintas. En el cronograma de la figura 1 vemos una representación gráfica de este sistema de transmisión.

Y hasta aquí todo lo referente a la primera parte

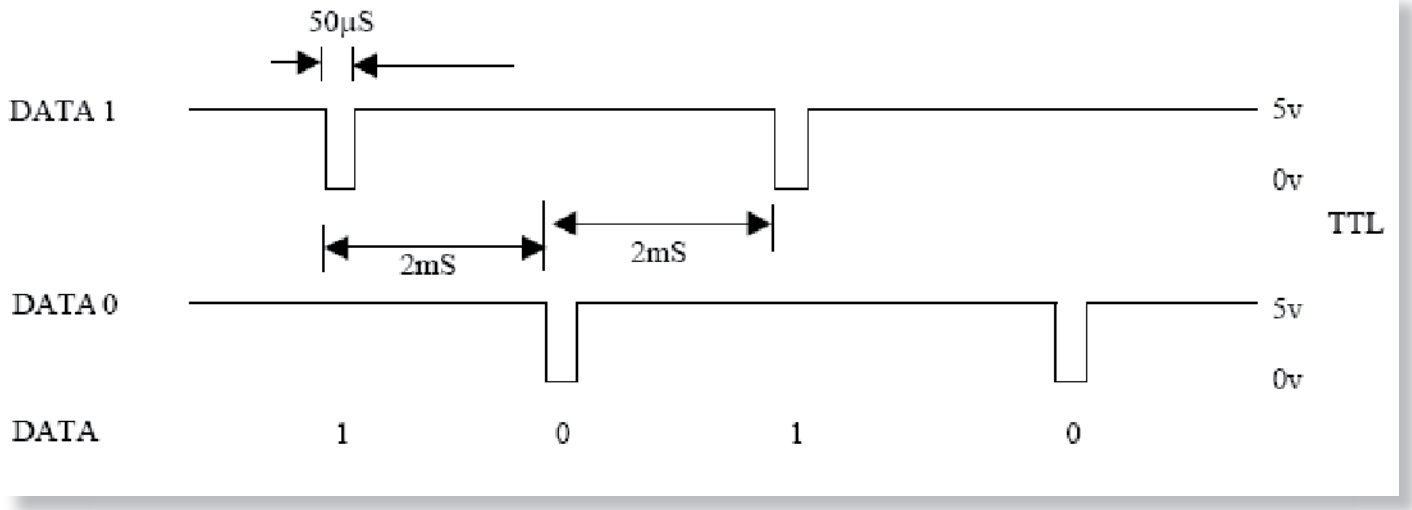


Diagrama de tiempos al transmitir la secuencia de bits 1010.

a la que nos referimos mas arriba como el modo en que físicamente se transmite la información digital.

.Interpretación de los Datos

Mediante el sistema descrito anteriormente se puede transmitir cualquier número de bits que queramos. Sin embargo, existe un cierto consenso para utilizar un determinado números de bits: 26 (el más utilizado), 32, 44 ó 128. Y la interpretación de los mismos, salvo el de 26 bits, es tan diversa como fabricantes lo utilizan.

Vamos a analizar en detalle el Wiegand 26, ya que es el formato de trama mas utilizado con diferencia. Su interpretación es como sigue:

- El primer Bit, B0, es la paridad Par de los primeros 12 bits transmitidos (B1:12).

- Los 8 siguientes, B1:B8 son un Byte, un Entero de 8 bits, al que llaman Facility Code.

- Los 16 siguientes: B9:B24 son dos Bytes, un Entero de 16 Bits, al que llaman User Code

- El último bit, B25, es la paridad Impar de los últimos 12 bits transmitidos (B13:24).

Curioso ¿verdad? En la figura 2 podéis ver una típica interpretación de un código Wiegand 26. Ese ejemplo constituye el Facility Code + User Code 4-24610. La paridad E es 1 para hacer Par la secuencia de 00000100011 que tiene tres unos y la paridad O es también 1 para hacer impar la secuencia 0000000100010 que sólo tiene dos unos.

Los demás tipos de Wiegand's tienen interpretaciones distintas, así el Wiegand 32 no lleva paridad y tanto el Facility Code como el User Code son dos números enteros de 16 bits. El Wiegand 44 es mas simpático aún si cabe: los 8 primeros bits son el Facility Code, los 32 siguientes son el User Code y los 4 últimos son el OR EXCLUSIVO de los 40 bits anteriores tomados de 4 en 4. Sorprendente, pero cierto.

.El programa

Para finalizar, proponemos una serie de rutinas en CCS para utilizar el protocolo Wiegand. Hemos comentado el código lo suficiente como para que sea fácilmente comprendido. Básicamente, se lee el código Wiegand y se vuelca lo leído sobre el canal serie:

Wiegand 26, secuencia de bits:

E	(b0 --------- b11)	(b12 ------- b23)	O
1	00000100	01100000100010	1

Donde E es paridad PAR para los bits 0 a 11 e IMPAR para los bits 12 a 23.

Interpretación de un código Wiegand 26.

```
// reader_wiegand26
//
// hardware MAHSA GP-20 Proximity cards
//
// red   -> vcc   -> +12V
// green -> data0 -> rb0 -> con-ml-10 -> 6
// white -> data1 -> rb1 -> con-ml-10 -> 4
// black -> gnd   -> gnd -> con-ml-10 -> 2

#include <18f4550.h>
#fuses HS,MCLR,NOWDT,NOPROTECT,NOPUT,NOBROWNOUT,NOPBADEN,NOLVP,NOCPD,NODEBUG,NOWRT,NOVREGEN
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#include <string.h>

#bit PIN_DATA0 = 0xF81.0
#bit PIN_DATA1 = 0xF81.1

const int CodeBits = 26;

int1 first_exp=0;
int1 read_complete=0;
int nextbit=0;
char Code[CodeBits+1];
int FacilityCode;
long IdCardCode;
char bits[5]={0x01,0x02,0x04,0x08,0x10};
int i;

// INTERRUPCION por EXT0 Data0 ------------------------------------------------
#int_ext
```

```
ext_handler() {

  if(first_exp==1){
    Code[nextbit]='0';
    if(++nextbit==CodeBits){
      read_complete=1;
    }
  }
  first_exp=1;
}

// INTERRUPCION por EXT1 Data1 ------------------------------------------------

#int_ext1
ext1_handler() {

  if(first_exp==1){
    Code[nextbit]='1';
    if(++nextbit==CodeBits){
      read_complete=1;
    }
  }
  first_exp=1;
}

void limpia_data(void){

  for(i=0;i<CodeBits+1;i++) Code[i]=0x00;
  nextbit=0;
}

void ajusta_code(void){

  // Finaliza Code
  Code[CodeBits]=0x00;
  // Facility Code
  FacilityCode=0x00;
  for(i=1;i<9;i++){
    if(Code[i]=='1'){
      FacilityCode=FacilityCode+bits[i-1];
    }
  }
  // Id Card Code
  IdCardCode=0x00;
  for(i=9;i<CodeBits;i++){
    if(Code[i]=='1'){
      IdCardCode=IdCardCode+bits[i-9];
    }
  }
}

void main() {

  disable_interrupts(global);
  set_tris_b(0b00000111);
  printf("Wiegand26 Reader listen\r\n\n");

  limpia_data();
  ext_int_edge(0,L_TO_H);
  ext_int_edge(1,L_TO_H);
  first_exp=0;

  enable_interrupts(int_ext);
  enable_interrupts(int_ext1);
  enable_interrupts(global);

  do{
    if(read_complete==1){
      read_complete=0;
      disable_interrupts(global);
      ajusta_code();
      printf("Facility Code = %u IdCardCode = %lu\r\n\n",FacilityCode,IdCardCode);
      limpia_data();
      enable_interrupts(global);
    }
  } while (TRUE);
}
```

Referencias:

Efecto Wiegand: http://en.wikipedia.org/wiki/Wiegand_effect

//Ariel Palazzesi//
arielpalazzesi@gmail.com

primera parte

PICs y LEDs: una pareja perfecta

Los diodos LED son seguramente la forma más popular de señalización de estados en los equipos electrónicos. En este pequeño artículo, veremos la manera de sacar todo el provecho posible a estos coloridos componentes.

En 1921 el físico Albert Einstein recibió el premio Nobel de esa ciencia. Pero contrariamente a lo que muchos creen no fue por su teoría de la relatividad, sino por un estudio en apariencia mucho más modesto: el efecto fotoeléctrico.

Einstein explicó que algunos materiales, al ser expuestos a una fuente de luz con determinada longitud de onda, se inducía una pequeña corriente eléctrica. También demostró que al hacer circular por ellos una corriente eléctrica, emiten luz.

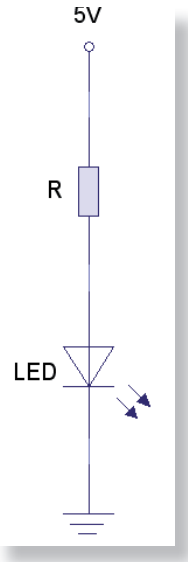
La luz producida mediante el llamado efecto fotoeléctrico tiene una frecuencia determinada (es de un sólo color), que depende del tipo de material. Algo parecido a lo que ocurre en un rayo láser, pero sin la coherencia que presenta el haz de luz de este ultimo.

Gracias el efecto fotoeléctrico el estadounidense Nick Holonyak, Jr. Inventó el primer dispositivo semiconductor que hacía uso práctico de este importante descubrimiento físico. Como en electrónica todo parece tener una sigla por nombre, se los llamo LED: Light Emiting Diode, o Diodo Emisor de Luz en español. Presente en forma de luz piloto en casi todos los componentes electrónicos de consumo, es la aplicación por excelencia de un efecto físico que permite la emisión de luz (fotones) cuando se recombinan un electrón y un hueco dentro de la unión PN que forma el diodo.

Este tipo particular de diodo se encapsula, generalmente, en plástico transparente, de manera que esta radiación sea visible. De acuerdo a los materiales utilizados en su fabricación, la luz emitida es de diferentes colores, siendo los más frecuentes el rojo, verde y amarillo, aunque es posible encontrarlos en gran variedad de colores, incluso blancos. Y por supuesto, algunos son capaces de emitir luz en una frecuencia que está más allá del color rojo, típicamente en 940 nanómetros, banda denominada infrarrojo y que se emplean como emisores en aparatos de control remoto o como barreras luminosas en tareas de automatismo y control.

Para que todo esto funcione, necesitamos que una corriente atraviese el LED. La intensidad de esta corriente debe ser cuidadosamente calculada, dado que si excedemos los límites especificados en la hoja de datos del componente, este se destruirá. La lista siguiente nos da una idea de que tensión aproximada necesita la juntura de los LEDs de colores comunes para funcionar:

- *Rojo = 1,6 V*
- *Rojo alta luminosidad = 1,9v*
- *Amarillo = 1,7 V a 2V*
- *Verde = 2,4 V*
- *Naranja = 2,4 V*
- *Blanco brillante= 3,4 V*
- *Azul = 3,4 V*
- *Azul 430nm= 4,6 V*



Una resistencia en serie con el LED limita la corriente que lo atraviesa.

Para limitar la corriente que circula por el LED, lo más usual, es colocar un resistor en serie con él.

El cálculo del valor de esta resistencia es muy sencillo, y solo implica el uso de la ley de ohm. Debemos restar la tensión del LED a la tensión de la fuente, y dividir el resultado por la corriente que deseamos atraviese el componente. Si usamos las unidades correctas (tensiones en Volts y corrientes en Amperes), el resultado estará expresado en Ohms.

$$R = \frac{V - V(LED)}{I(LED)}$$

Esta fórmula permite calcular el valor de la resistencia limitadora.

Veamos un ejemplo concreto. Supongamos que tenemos un LED rojo de alta luminosidad, que según su hoja de datos, necesita para funcionar correctamente, una corriente de 18 mA y una tensión entre ánodo y cátodo de 2 V, y queremos alimentarlo con una batería de 9V ¿Cuál será el valor de la resistencia limitadora?

Bien, si aplicamos la formula anterior, obtenemos que

$$R = \frac{V - V(LED)}{I(LED)}$$

Reemplazamos los valores, y calculamos R.

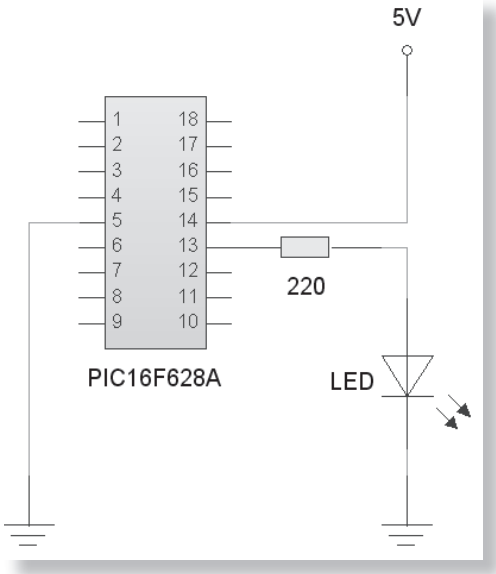
Como puede verse, el valor para la resistencia es de 389 Ohms. Como no existen resistores comerciales de ese valor, utilizaremos el más cercano: 390 Ohms.

A menudo es necesario colocar dos o más LEDs en serie, entre sí. En ese caso, debemos asegurarnos que todos funcionan con la misma corriente, para evitar que alguno resulte dañado. Luego, simplemente reemplazamos en la formula que hemos visto el valor de V(LED) por la suma de las tensiones de cada uno de los LEDs implicados. Por supuesto, el valor de esta suma no debe ser mayor a la tensión suministrada por la fuente.

.LEDS y PICS

Por supuesto, los LEDs resultan ser la lámpara ideal para los microcontroladores. Su bajo consumo de corriente hace que puedan manejarse directamente con sus pines (casi siempre) sin necesidad de etapas intermedias. Pero antes de realizar cualquier diseño electrónico, debemos asegurarnos que la corriente suministrada por las salidas del microcontrolador en cuestión es suficiente para el modelo de LED que vamos a emplear.

La figura siguiente ilustra la forma en que podemos conectar un LED a un PIC. Vamos a suponer que la tensión disponible en el pin del PIC es de 5V, que la corriente que atravesara el LED es de 15 mA, y que la caída de tensión en este es de 2V. Eso nos da un valor para R de 200 Ohms, por lo que utilizaremos una de 220 Ohms, valor disponible comercialmente, más cercano.



LED conectado entre PORTB.7 y GND.

"Los LEDs resultan ser el complemento ideal para los microcontroladores"

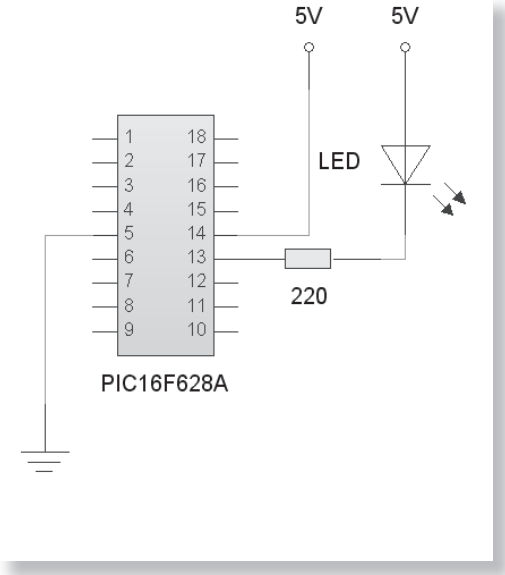
El siguiente programa en CCS permite probar el funcionamiento del circuito de ejemplo:

```
#include <16f628a.h>  //PIC utilizado
#use delay (clock=4000000)    //Oscilador a 4Mhz
#use fast_io(b)   //Optimizamos E/S del PORTB
//------Programa principal---------------------
void main(void)
{
   set_tris_b(0x7F);    // RB7como salida,
                           el resto como entrada.
   do{
      output_low(PIN_B7);    //Apago el LED
      delay_ms(500);         //Espero 500ms.
      output_high(PIN_B7);   //Enciendo el LED
      delay_ms(500);         //Espero 500ms.
   }while(TRUE);             //Repito el bucle
}
```

Y el siguiente hace lo mismo, pero está escrito en el BASIC del PIC SIMULATOR IDE:

```
AllDigital          'Desabilitamos comparadores
TRISB = %01111111  'RB7como salida, el resto como
entrada.
 loop:
      PORTB.0 = 0    'Apago el LED
      WaitMs 500     'Espero 500ms.
      PORTB.0 = 1    'Enciendo el LED
      WaitMs 500     'Espero 500ms.
 Goto loop
END
```

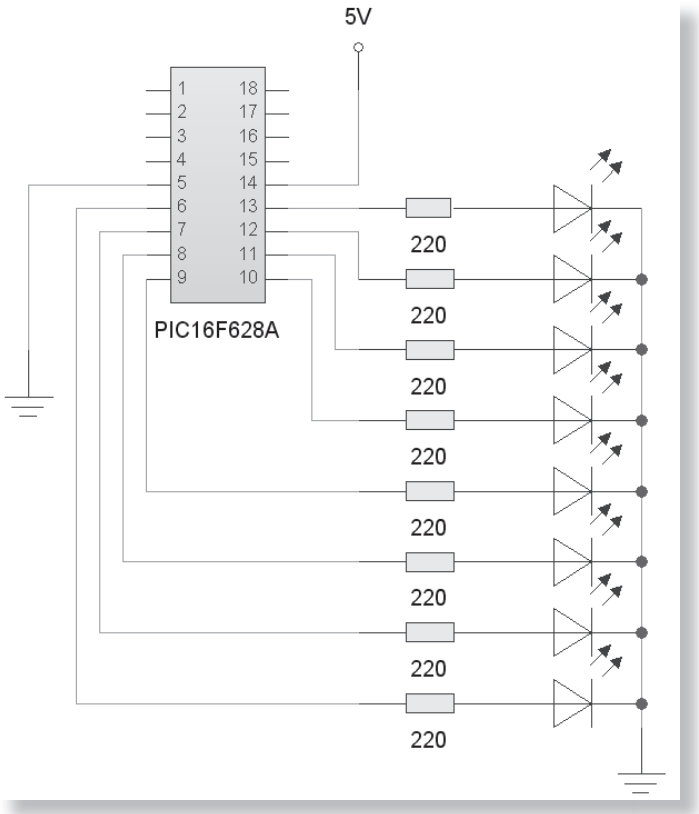
En el ejemplo anterior, el LED encenderá cuando el pin 7 del puerto B se ponga en nivel alto. Pero también podemos conectar el LED de manera que se encienda al poner el pin en nivel bajo:



LED conectado entre PORTB.7 y Vcc.

Si probamos cualquiera de los dos programas precedentes con este circuito, también funcionará, solo que el LED encenderá cuando debería estar apagado, y viceversa.

Nada impide manejar más de un LED a la vez. De hecho, se trata simplemente de repetir el esquema "resistencia limitadora – LED" tantas veces como sea necesario:



8 LEDs conectados al puerto B del PIC16F628A

El código siguiente, en el mismo BASIC que usamos antes, permite realizar una cuenta en binario desde 0 a 255, al tiempo que muestra el resultado en los 8 LEDS conectados al puerto B:

```
AllDigital          'Desabilitamos comparadores
TRISB = %00000000   'Todo el puerto b como salida.
Dim i As Byte

For i = 1 To 255
PORTB = i           'Enciendo leds
WaitMs 500          'Espero 500ms.
Next i
End
```

También podemos ver un ejemplo en CCS. El programa siguiente consiste en un bucle infinito, que en cada repetición cambia el estado de las salidas del puerto B, encendiendo uno de los LEDs a la vez. Cuando llega a uno de los pines del extremo del puerto, realiza el efecto inverso hasta llegar nuevamente al otro. Luego se repite todo el ciclo. Se han empleado los operadores "<<" y ">>" para desplazar el contenido del puerto en uno y otro sentido.

```
#include <16f8627a.h>          //PIC utilizado
#use delay (clock=4000000)     //Oscilador a 4Mhz
#use fast_io(b)                //Optimizamos E/S
del PORTB
#byte portb = 0x6              //Direccion del
PORTB
//------Programa principal-----
void main(void)
{
   int i;                      //Declaro la va-
riable del for
   set_tris_b(0x00);           //Todo PORTB como
salida.
   disable_interrupts(GLOBAL);//todas las
interrupciones desactivadas
   do{                         //Bucle infinito
      portb = 0x01;  //Estado inicial del puerto
//----Bucle de "ida" ---------
      for (i=0;i<7;i++) {     //i cuenta de 0 a 6
        portb = portb << 1;   //Desplazo valor
de PORTB una pos. a la izq.
        delay_ms(500);//Espero medio segundo y..
      }
      //----Bucle de "regreso" -----
      for (i=7;i>0;i--) {     //i cuenta de 7 a 0
        portb = portb >> 1;       /Desplazo valor
de PORTB una pos. a la der.
        delay_ms(500) //Espero medio segundo y...
      }
   } while(TRUE);            //...repito el bucle
}
```

Hay casos en que se necesitan manejar un número elevado de LEDs, y los pines disponibles en el microcontrolador de turno resultan insuficientes. La solución es utilizar alguna técnica de multiplexado. En el próximo número veremos cómo hacerlo. ■

//Jesús Fernández Hernández//
jfh900@hotmail.com

poniendo un poco de orden en "C"

Mantener ordenadas aquellas librerías que compartimos entre distintos proyectos, puede ser una tarea a veces complicada, sobre todo cuando desconocemos métodos eficaces para realizar este trabajo. Este problema existe desde hace muchos años y los desarrolladores de compiladores para C, fueron incluyendo mecanismos eficientes para dar solución a este problema. Las siguientes líneas nos ayudarán a sacar provecho de esas simples, pero poderosas herramientas.

El uso de las librerías es fundamental para el desarrollo de proyectos en "C". Sin embargo, cuando tenemos varios proyectos que comparten las mismas librerías, una gestión deficiente, puede llevarnos al caos.

¿Donde reside el problema?

En aquellas librerías que necesitamos reutilizar, y que por su naturaleza, tenemos que modificar para adaptarlas a nuestros proyectos. Un claro ejemplo de este tipo de librerías es la Flex_LCD.c desarrollada por CCS, que nos permite utilizar los muy comunes LCD's de 2 líneas. Habitualmente, esta librería debe ser modificada para adaptarla a nuestras necesidades en cada proyecto. Esta situación se presenta cuando nuestros proyectos requieren el uso de distintos microcontroladores o cuando necesitamos determinados módulos del microcontrolador, cuyos pines de E/S, han sido asignados al LCD dentro de Flex_LCD.c. De aquí en adelante, utilizaremos la librería "Flex_LCD.c" como modelo para el resto del artículo, pero todo lo expuesto es aplicable cualquier librería.

> **"Estas directivas son una herramienta muy poderosa para crear variables, reservar memoria, etc."**

¿Como se modifican estas librerías para su uso?

Aquí es donde surge el caos entre los distintos proyectos que tenemos entre manos o que hemos realizado. Analicemos las tres alternativas, de uso más frecuente:

.La forma usual e ineficaz.

Tenemos una única librería ubicada en el directorio de librerías (library), y cuando nos hace falta, la modificamos. Esta suele ser una práctica muy habitual. Cada vez que empezamos un nuevo proyecto modificamos la librería y la adaptamos a la necesidad del momento

Pero: ¿qué ocurre cuando debemos modificar y recompilar un proyecto hecho con anterioridad? Si los pines utilizados en el proyecto anterior y el actual coinciden, no tendremos problema alguno. Sin embargo, es frecuente que no coincidan los pines asignados al LCD del antiguo proyecto con los del actual. Por lo que si compilamos un proyecto antiguo, es muy probable que no funcione correctamente.

La solución común al problema anterior, es tener anotado en algún lugar la asignación de pines para cada proyecto y modificar la librería antes de compilar cada uno. Como se pude ver, es un proceso tedioso que exige un alto grado de orden para mantener la funcionalidad de nuestros proyectos.

.El método de la copia

Una alternativa que puede solucionar el problema anterior, es tener una copia de la librería en el directorio de cada proyecto. Luego modificamos la copia, para ajustarla a la configuración según sea el caso. Esto permite que podamos compilar cada proyecto una y otra vez, sin necesidad de modificar la librería, ya que cada proyecto tiene una copia adaptada según sus necesidades.

Es una solución también bastante habitual, pero no idónea; ¿qué ocurre si necesitamos modificar la librería porque tenemos una nueva versión de la misma? Tendremos que ir buscando por el laberinto de directorios de proyectos cada copia de la librería vieja y sustituirla por la nueva.

Se puede argumentar que hoy en día con la velocidad de proceso y las herramientas de búsqueda de las PC, este trabajo no será en extremo tedioso. Pero aunque lográsemos encontrar y sustituir todas las copias en un corto espacio de tiempo, tendremos otro problema añadido, y es que cada copia de la librería está "personalizada" para su proyecto. La situación anterior nos obliga a reconfigurar la nueva versión de la copia, de acuerdo a la configuración

de cada proyecto, trabajo que hicimos la primera vez que copiamos la librería hacia el directorio del proyecto.

.Utilizando las directivas del pre-procesador

Esta es la forma correcta y eficaz de hacerlo. Este método es el que adoptaremos y nos permitirá manejar las librerías sin sufrir dolores de cabeza. Consiste en definir la asignación de pines, en algún lugar fuera de la librería, bien en fichero aparte, o bien en el programa principal del proyecto. ¿Cómo podemos modificar la asignación de pines fuera de la librería? La forma de hacerlo es utilizando las directivas del pre-procesador.

Las directivas del pre-procesador son un conjunto de instrucciones que se utilizan para indicarle al compilador, que debe hacer, ante determinadas situaciones. Aunque generalmente muchos programadores desconocen su utilidad con profundidad, estas directivas son una herramienta muy poderosa para crear variables, reservar memoria, definir constantes, utilizar macros e incluso indicarle al compilador que secciones de código debe compilar y enlazar. En nuestro caso, utilizaremos las directivas del pre-procesador `#ifndef <identifier> … #endif.`

Cuando el pre-procesador se topa con la directiva `#ifndef`, comprueba si ya existe el identificador `<identifier>`, si éste no existiese, entonces crea uno con ese nombre, lo agrega a su lista de identificadores y procesa el código ubicado entre `#ifndef` y `#endif`, en caso que el identificador `<identifier>` exista, se ignora todo el código ubicado en el cuerpo de la llamada a la directiva.

> **"Estas directivas son una herramienta muy poderosa para crear variables, reservar memoria, etc."**

La técnica descrita anteriormente es precisamente la que vamos a utilizar para gestionar de manera eficiente, el uso de nuestras librerías. Al revisar la sección de **Flex_LCD**, donde se asignan los pines al microcontrolador, nos topamos con el siguiente código:

```
#define LCD_DB4   PIN_B4
#define LCD_DB5   PIN_B5
#define LCD_DB6   PIN_B6
#define LCD_DB7   PIN_B7
#define LCD_RS    PIN_C0
#define LCD_RW    PIN_C1
#define LCD_E     PIN_C2
```

Ahora simplemente metemos esta sección de código en el cuerpo de una llamada a `#ifndef` con nombre de identificador `_FLEX_LCD`, el código resultante quedará de la siguiente forma:

```
#define _FLEX_LCD
    #define LCD_DB4   PIN_B4
    #define LCD_DB5   PIN_B5
    #define LCD_DB6   PIN_B6
    #define LCD_DB7   PIN_B7
    #define LCD_RS    PIN_C0
    #define LCD_RW    PIN_C1
    #define LCD_E     PIN_C2
#endif
```

Si no definimos nada en el programa principal o en su fichero de cabecera, el pre-procesador asignará a la LCD los pines según el código de la librería **Flex_LCD**. Si queremos modificar la asignación de pines para nuestro proyecto, escribiremos en el fichero principal de nuestro proyecto, o en su fichero de cabecera, el siguiente fragmento de código:

```
#define _FLEX_LCD
#define LCD_DB4   PIN_C4
#define LCD_DB5   PIN_C5
#define LCD_DB6   PIN_C6
#define LCD_DB7   PIN_C7
#define LCD_RS    PIN_A0
#define LCD_RW    PIN_A1
#define LCD_E     PIN_A2
#include Flex_LCD.c
```

Esto hace que se asignen los pines del microcontrolador a la LCD tal y como se especifica en nuestro programa principal y que la definición de la librería sea ignorada. Como puede verse, la librería ha sufrido un pequeño cambio que nos ayudará a mantener gestionado su uso y nos facilitará la vida a partir de este momento. Es muy importante que esta asignación se haga antes de incluir la librería (`#include Flex_LCD.c`), ya que de no hacerlo así, el pre-procesador asignará los pines según la definición que se hace dentro de la librería y se producirá un conflicto con la definición realizada en el programa principal.

Con este método, solo tendremos una librería para todos nuestros proyectos y la personalización se realizará dentro de cada proyecto; sin que por ello tengamos que hacer copias o modificar el fichero original. Además, la librería estará perfectamente localizable dentro de su directorio, por lo que si obtuviésemos una nueva versión, bastará con actualizar y modificar una sola copia.

Otra razón para utilizar esta forma de proceder, es la posibilidad de reconocer la dependencia entre los distintos archivos de nuestros proyectos o entre distintas librerías. Por ejemplo, si creamos una librería que utilice el display como salida, podremos escribir en el código de nuestra librería:

```
#ifndef _FLEX_LCD
   #error Es necesario incluir la librería Flex_LCD
#endif
```

De esta forma enviamos un mensaje de error para avisar que es preciso incluir una o varias librerías. ■

//Ariel Palazzesi//
arielpalazzesi@gmail.com

simulador de circuitos digitales

El nombre completo del software que vamos a analizar hoy es Simulador de Construcción de Circuitos Digitales con Escenarios Virtuales y Tutoriales Interactivos. Como su nombre lo indica, es un programa que nos permite construir y evaluar circuitos digitales utilizando para ello un módulo digital virtual.

Todos los que alguna vez hemos diseñado un circuito digital sabemos de lo importante que es contar con una herramienta que nos permita llevar a cabo una simulación del funcionamiento del circuito en que estamos trabajando. Estas herramientas suelen ser bastante caras, y muchas veces el hobbysta o aficionado a la electrónica no tiene acceso a ellas. Afortunadamente, existe el Simulador de Construcción de Circuitos Digitales con Escenarios Virtuales y Tutoriales Interactivos ("el Simulador", de aquí en adelante), que es completamente gratis y no tiene nada que envidiarle a muchas de las herramientas de pago.

Este programa ha sido desarrollado por Arturo Javier Miguel De Priego Paz Soldán, Ingeniero Electrónico de la Pontificia Universidad Católica del Perú. La versión que evaluaremos es la 0.94, la última disponible al momento de escribir este articulo. El programa corre bajo Windows, y necesita de una resolución de pantalla de por lo menos 1024 x 768 píxeles.

.El programa

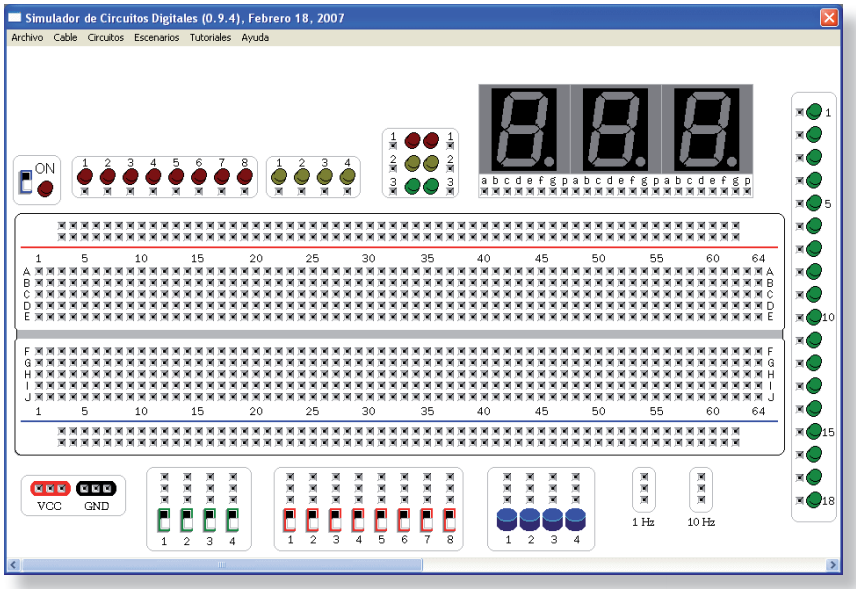
El software permite construir y simular circuitos digitales, a partir de modelos lógicos de circuitos integrados estándares (de la familia TTL "LS") y de aplicación específica (conocidos como ASIC). Los circuitos que construye el usuario pueden ser simulados directamente sobre el módulo digital que provee el programa o, en algunos casos, ser validados sobre los Escenarios Virtuales. Estos escenarios representan el entorno en el que los circuitos operarán. Por supuesto, los esquemas construidos pueden ser almacenados, recuperados y editados.

El programa también provee una serie de Tutoriales Interactivos, que se encargan de ilustrarnos sobre el funcionamiento de algunos circuitos lógicos típicos. Muchos de ellos incluyen descripciones VHDL. Según nos cuenta Arturo, "el software ha sido diseñado para ser empleado como una herramienta de enseñanza y aprendizaje del diseño digital, y actualmente está orientado a los cursos básicos o de introducción a los circuitos digitales, tanto en el nivel escolar como universitario."

Entre los puntos sobresalientes de este programa, podemos destacar el gran número de modelos de circuitos integrados TTL que el autor ha incluido (y sigue incluyendo). La posibilidad de almacenar y recuperar nuestros proyectos permite la verificación y reutilización de los mismos, tanto en la enseñanza como en el aprendizaje del diseño digital. La existencia de los tutoriales, que se muestran a la derecha del módulo digital permite validar rápidamente el conocimiento adquirido. Los escenarios, aunque pocos por ahora (el programa se encuentra en permanente desarrollo), nos brindan una mejor perspectiva y facilitan una mejor primera especificación del diseño lógico.

La inclusión de módulos ASIC simplifican los diseños, a la vez que ahorran espacio en el protoboard virtual. El usuario puede crear nuevos modelos de ASIC, a partir de descripciones VHDL o programas C++.

El autor comenta que todavía resta trabajo por hacer, para que el producto sea aun más flexible. Por ejemplo, no es posible por ahora que el usuario diseñe nuevos modelos de circuitos integrados TTL para agregarlo a la

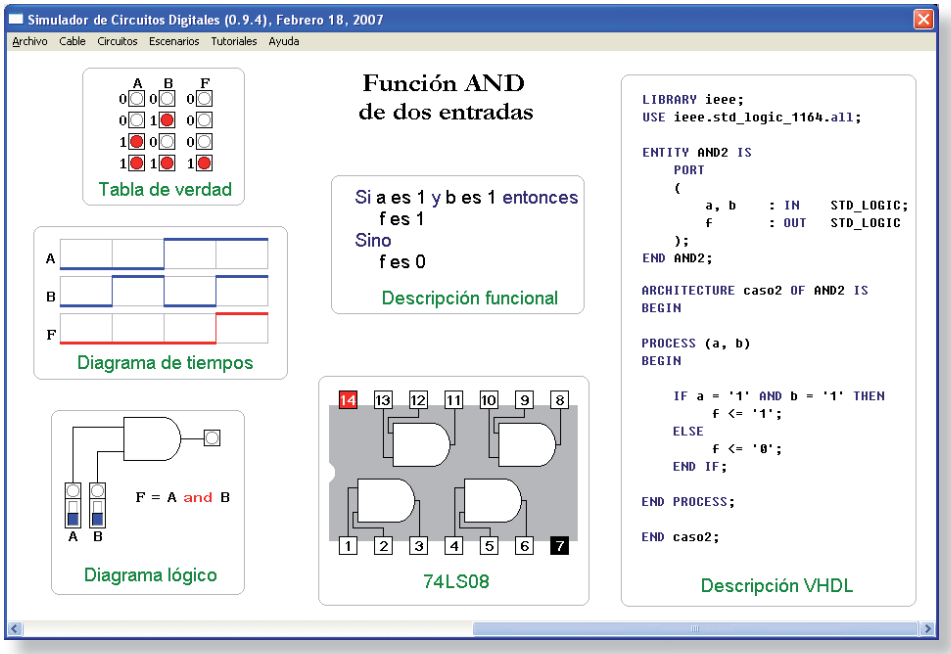


Epígrafe: Pantalla principal del programa, mostrando el Módulo Digital.

biblioteca del programa. Esto será subsanado en las próximas versiones, las que permitirán usar VHDL también para esta tarea.

Tampoco se han considerado los efectos eléctricos (retardos en la propagación de las señales, abanicos de entrada y salida, ruido, etc.), y los chips modelados no cuentan con pines de tres estados ni bidireccionales.

Esto no representa una limitación demasiado grabe. El simulador, tal como se encuentra en la actualidad, es perfectamente operativo y cumple a la perfección su misión educativa. Cuenta con cuatro modelos de ASICs y casi 100 integrados TTL listos para usar.

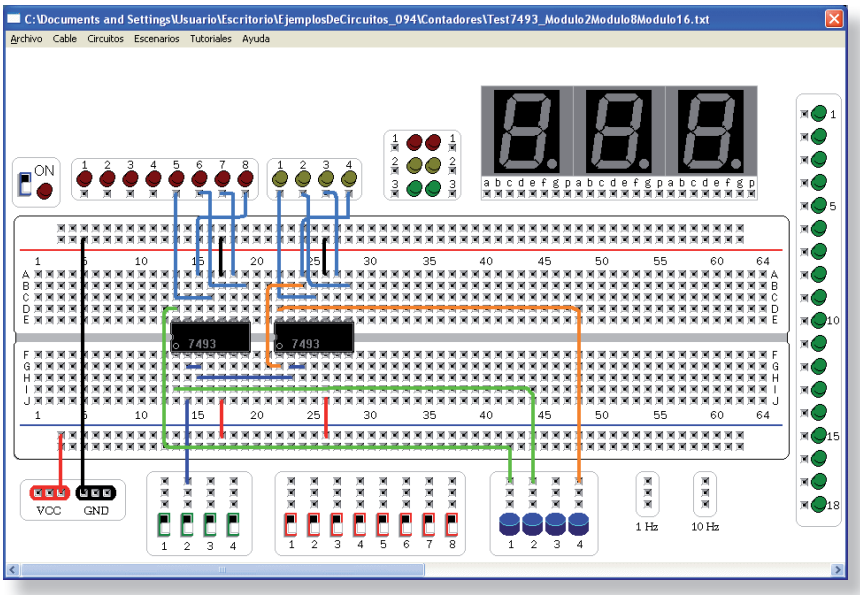


Ejemplo del tutorial correspondiente a la puerta AND

.El Módulo Digital

De alguna manera, este es el corazón del programa. Sobre el, el usuario va desplegando los componentes y conexiones que conforman su proyecto. Esta compuesto por una tarjeta para alambrar circuitos (también conocida como protoboard o breadboard), 18 LEDs, 3 visualizadores de siete segmentos, generadores de reloj, entradas digitales (12 interruptores y 4 pulsadores), bornes de alimentación (VCC y GND), una bornera de expansión de 18 líneas (a la que se conectan los escenarios virtuales) y un interruptor principal para el encendido y apagado del sistema

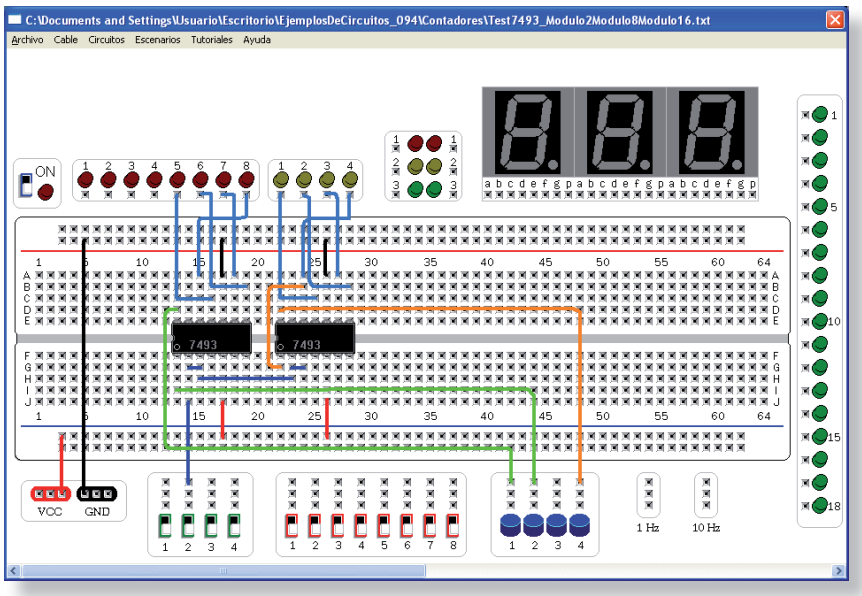
Para montar un circuito, simplemente vamos seleccionando los chips necesrios desde un menú (que los agrupa por categorías) y los insertamos sobre el protoboard. Las conexiones entre ellos se dibujan trazando líneas con el ratón. De ser necesario, tanto los cables como los chips pueden retirarse simplemente pulsando con el botón derecho sobre el chip en cuestión o sobre uno de los extremos del cable a remover.



Circuito de prueba, basado en un contador con puertas lógicas simples.

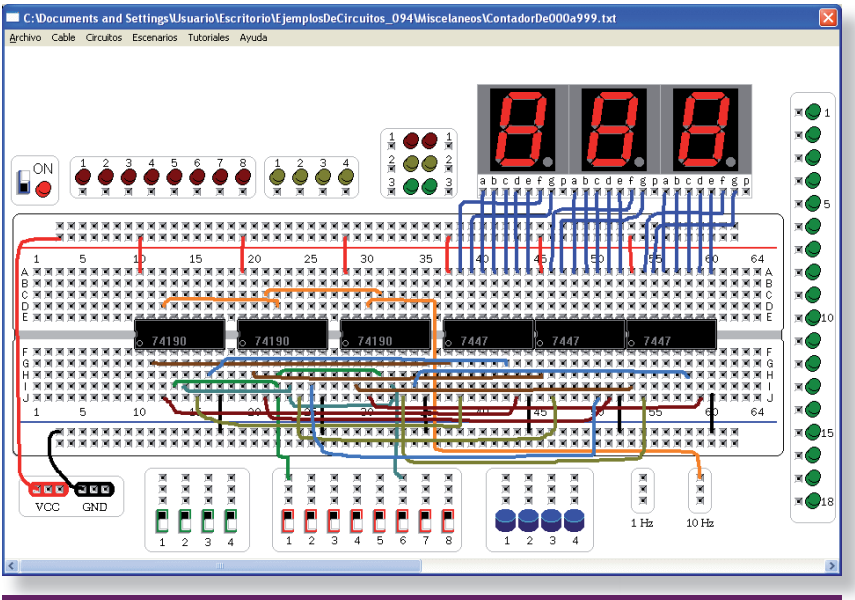
.Escenarios y tutoriales

Como decíamos, el programa proporciona escenarios sobre los que probar nuestros circuitos. Estos pueden elegirse e insertarse desde un menú. Cuando el interruptor principal se encuentra en la posición de “apagado”, el escenario trabaja en modo independiente, siguiendo un comportamiento predefinido. En este modo el usuario puede observar cómo debe interactuar el circuito con el medio ambiente virtual. Cuando el interruptor se enciende, el escenario se controla mediante las señales que provienen desde el Módulo Digital

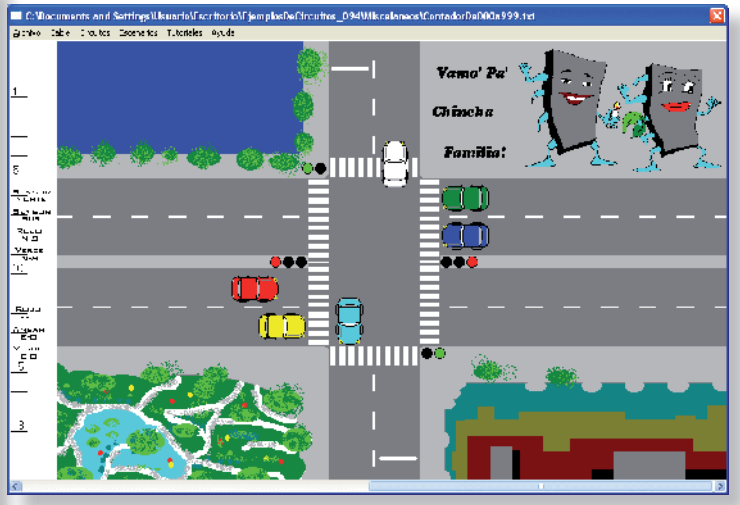


Decodificador binario y displays de siete segmentos.

Los tutoriales nos muestran los aspectos bá-



Contador BCD con habilitación de cuenta ascendente/descendente, de 0 a 999.



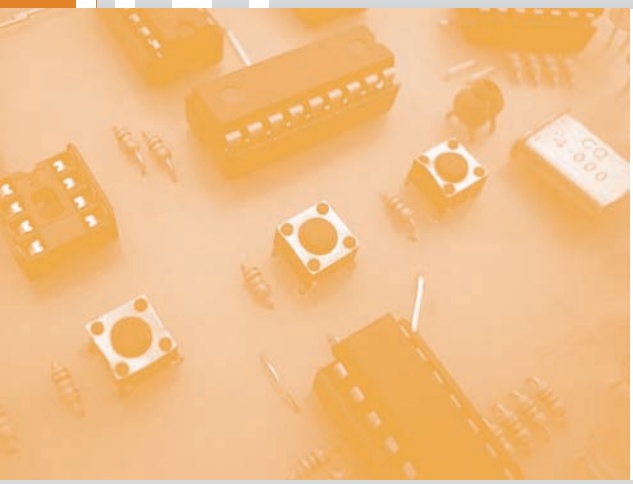
Escenario de un cruce de avenida con calle de bajo tránsito.

sicos de algunos temas relacionados con la electrónica digital. En muchos casos podemos ver la descripción VHDL correspondiente. Los hay enfocados al funcionamiento de las puertas básicas, descodificadores, multiplexores, sumadores, comparadores, latches y flipflops.■

Referencias:

El programa puede ser descargado desde SimuladorDigital_094.zip
Una guía preliminar está en GuiaPreliminar_094.zip
Ejemplos de algunos circuitos están en EjemplosDeCircuitos_094.zip
Correo electrónico: amiguel@pucp.edu.pe
Página WEB: www.geocities.com/tourdigital

//Ariel Palazzesi//
arielpalazzesi@gmail.com



"el relojito"

segunda parte

En esta segunda entrega analizaremos a fondo las rutinas necesarias para la programación de nuestro reloj. Utilizaremos los lenguajes PIC BASIC y CCS, lo que permitirá a una gran cantidad de lectores comprender el funcionamiento de este proyecto.

En el número anterior de la revista vimos como desarrollar el hardware necesario para la construcción de un completo reloj digital, que además de la hora, era capaz de mostrar la temperatura ambiente gracias a la inclusión de un sensor de temperatura Dallas DS1820. También se incluía un RTC ("Real Time Clock", o "Reloj de Tiempo Real") DS1307, encargado de proporcionar al PIC16F628A la necesaria y precisa referencia temporal. Un grupo de 60 diodos LED, indicando cada uno de los segundos, dotaban al proyecto de un atractivo especial.

A partir de este numero, comenzaremos a ver como sacar provecho de este diseño, explicando cada una de las rutinas necesarias para transformar ese montón de hardware en algo útil. Escribiremos el software desde cero, explicando cada uno de los pasos a seguir, de forma que todos puedan comprender a fondo cada una de las rutinas, y así ser capaces de modificarlas a gusto. Hemos decidido presentar versiones en PIC BASIC y CCS de cada trozo de código, como una manera de llegar a un mayor numero de lectores.

En el número anterior de ***uControl*** vimos como desarrollar el hardware necesario para la construcción de un completo reloj digital, que además de la hora, era capaz de mostrar la temperatura ambiente gracias a la inclusión de un sensor de temperatura Dallas **DS1820**. También se incluía un RTC ("Real Time Clock", o "Reloj de Tiempo Real") **DS1307**, encargado de proporcionar al **PIC16F628A** la necesaria y precisa referencia temporal. Un grupo de 60 diodos LED, indicando cada uno de los segundos, dotaban al proyecto de un atractivo especial.

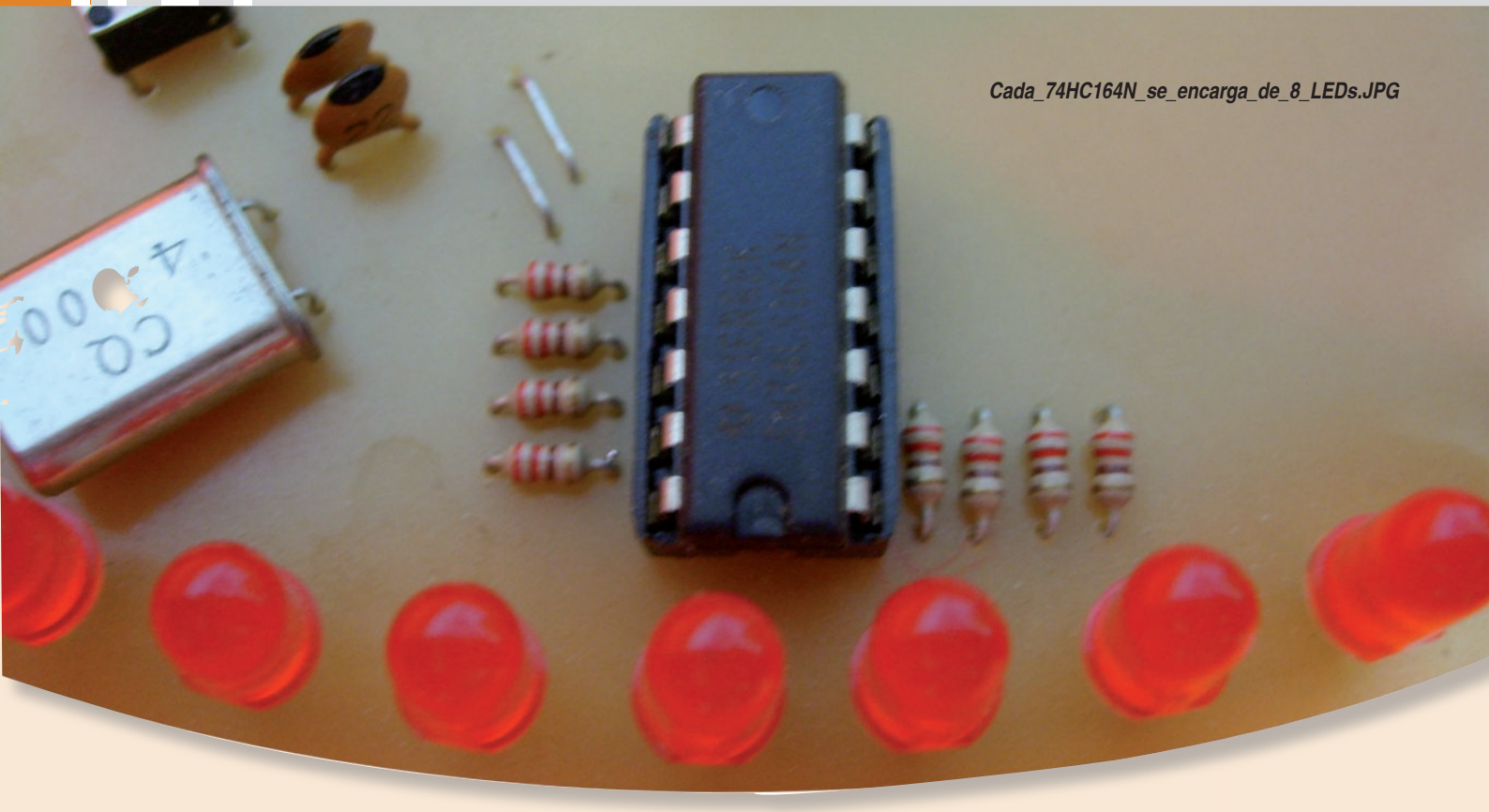
A partir de este número, comenzaremos a ver como sacar provecho de este diseño, explicando cada una de las rutinas necesarias para transformar ese montón de hardware en algo útil. Escribiremos el software desde cero, explicando cada uno de los pasos a seguir, de forma que todos puedan comprender a fondo cada una de las rutinas, y así ser capaces de modificarlas a gusto. Hemos decidido presentar versiones en PIC BASIC y CCS de cada trozo de código, como una manera de llegar a un mayor numero de lectores.

.Configuración de los pines de E/S

El primer paso antes de comenzar a utilizar los pines de entrada o salida, es configurar correctamente su función. Este es un paso muy simple, y que no requiere de demasiadas explicaciones. Los comentarios incluidos en el código fuente serán suficientes para entender que hace cada línea de programa.

Veamos primero como hacerlo en **PIC BASIC** (recordemos que utilizamos la versión correspondiente al **PIC SIMULATOR IDE):**

```
'-----CONFIGURAMOS PUERTOS----------
AllDigital  'Todos los pines del PORTA como E/S

'Configuro el PORTA:
TRISA.0 = 0  'DATA Segundero
TRISA.1 = 0  'CLOCK Segundero
TRISA.2 = 0  'DATA HH:MM
TRISA.3 = 0  'CLOCK HH:MM
TRISA.4 = 0  'Salida
TRISA.5 = 0  'DS1820
```



Cada_74HC164N_se_encarga_de_8_LEDs.JPG

```
'Configuro el PORTB:
TRISB.0 = 1  'Entrada pulsos del DS1307
TRISB.1  =  0   'Salida,  LEDs  ":"  en  display
"HH:MM"
TRISB.2 = 0  'pin SCA del DS1307
TRISB.3 = 0  'pin SCL del DS1307
TRISB.4 = 1  'Entrada Pulsador 1
TRISB.5 = 1  'Entrada Pulsador 2
TRISB.6 = 1  'Entrada Pulsador 3
TRISB.7 = 1  'Entrada Pulsador 4
```

Inicialización de puertos en PIC BASIC.

Ahora, vemos como hacer la misma tarea en CCS:

```
//Device/Fuses/Etc.--------------------------
#INCLUDE <16F628A.H>      //Usamos un 16F628A
#FUSES NOWDT              //No Watch Dog Timer
#FUSES XT                 //Con oscilador a
cristal...
#use delay(clock=4000000) //..de 4MHz.
#FUSES NOPUT              //No Power Up Timer
#FUSES NOPROTECT    //No protegemos el código.
#FUSES NOBROWNOUT   //No Brownout Reset
#FUSES NOLVP        //No low voltage prgming
#FUSES NOCPD        //No EE protection

//Declaramos la posición de los puertos-------
#BYTE PORTA = 0x05
#BYTE PORTB = 0x06
#BYTE PORTA_TRIS = 0x85
#BYTE PORTB_TRIS = 0x86

//Y asignamos cada pin como E/S según corresponda:
PORTA_TRIS = 0b00000000; //1=ENTRADA, 0=SALIDA
PORTB_TRIS = 0b11110001; //1=ENTRADA, 0=SALIDA
```

Inicialización de puertos en CCS

Una vez listo este tramite, pasemos a las rutinas propiamente dichas.

.Escribiendo los segundos.

La característica más sobresaliente de este reloj es su segundero. Compuesto por 60 LEDs ubicados sobre la circunferencia del reloj, se controlan mediante solo dos pines del microcontrolador. Esto es posible gracias a la utilización de un registro de desplazamiento.

Como ya hemos visto, este tipo de registro incorpora los datos presentes en su entrada con cada pulso de reloj que se aplica a su terminal **CLOCK**.
Debemos respetar los tiempos de respuesta de los circuitos integrados que conforman los registros de desplazamiento. En este caso, el 74HC164N que henos utilizado puede funcionar a una frecuencia mas elevada que los 4 MHz (o el MIP) a los que funciona el 16F628A, por lo que no serán necesarios los tiempos de espera entre el envió de un dato y el siguiente.

Los pines implicados en el control de los LEDs del segundero son los correspondientes a **PORTA.0** (o RA0, pin 17) y **PORTA.1** (o RA1, pin 18), para las funciones de **DATA** y **CLOCK** respectivamente. Como no es el único registro de desplazamiento presente en el proyecto, nos referiremos a estas señales como **DATA2** y **CLOCK2.**

Concretamente, el dato presente en la entrada del registro de desplazamiento se hace presente en la primera de sus salidas ("empujando" a los demás una posición hacia delante) cuando el pin CLOCK pasa de estado bajo a estado alto. Esto quiere decir que deberemos seguir el siguiente orden para cada bit que queramos enviar al registro:

1) *Poner el dato a enviar en el pin DATA2 (PORTA.0)*
2) *Poner CLOCK2 (PORTA.1) en estado bajo.*
3) *Poner CLOCK2 (PORTA.1) en estado alto.*

Como se ve, es algo muy sencillo de implementar. Veamos como hacerlo en PIC BASIC:

```
escribo_segundo:
      data1 = bit_aux  'Pongo el valor en DATA
      clock1 = 0  'Pongo el CLOCK en bajo...
      clock1 = 1  '...y de nuevo en alto. Listo!
Return
```

Esta rutina envía el valor de la variable "bit_aux" al registro de desplazamiento.

La subrutina supone que antes de llamarla hemos declarado las variables (usando DIM) y las "macros" (mediante SYMBOL) necesarias:

```
'------DECLARO VARIABLES y MACROS--------------
Dim bit_aux As Bit  'Declaro la variable auxiliar
Symbol data1 = PORTA.0  'Nos referimos a PORTA.0
como "data1"
Symbol clock1 = PORTA.1  'Nos referimos a PORTA.1
como "clock1"
```

Declaración de variables y macros necesaria antes de llamar a la subrutina "escribo_segundo".

Veamos la forma de hacer esto en CCS.

```
//Declaramos la Variable:
int1 bit_aux;         //Declaro la variable
auxiliar

//-----------------------------------------
//---Envia un DATO al registro de desplazamiento:
//-----------------------------------------
void escribo_segundo(int1){
   if (bit_aux) {output_high(DATA2);} //Si es
"1", lo escribo en DATA2.
   if (!bit_aux) {output_low(DATA2);} //Si es
"0", lo escribo en DATA2.
   output_low(CLOCK2);   //Pongo el CLOCK en
bajo...
   output_high(CLOCK2);  //...y de nuevo en
alto. Listo!
}

main(){
   //Asignamos cada pin como E/S según corres-
ponda:
   PORTA_TRIS = 0b00000000; //1=ENTRADA,
0=SALIDA
   PORTB_TRIS = 0b11110001; //1=ENTRADA,
0=SALIDA
}
```

En CCS, para hacer uso de la función "escribo_segundo", basta con invocarla desde el programa principal, de la siguiente manera:

```
      escribo_segundo (valor);
```

Donde "valor" será "0" o "1" dependiendo si queremos apagar o encender el LED correspondiente a la primera posición del registro de desplazamiento.

.Haciendo limpieza

Antes de comenzar a enviar datos útiles al registro de desplazamiento, conviene "limpiar" el contenido de sus 60 bits, dado que al alimentar ***El Relojito*** pueden contener información aleatoria, que en la practica se verían como una serie de LEDs encendidos. Si no lo hiciéramos, cada dato que enviemos al registro "empujaría" a los bit-basura una posición hacia delante, algo que no quedaría demasiado bien.

La forma de evitar esto es bien simple: ni bien comienza nuestro programa, debemos escribir 60 ceros en el registro de desplazamiento, asegurándonos que todos los LEDs se encuentran apagados.

Dado que puede se trata de una acción que puede requerirse mas de una vez en nuestro programa, también la vamos a implementar como una subrutina (en PIC BASIC) o como una función (en CCS). Dado que ya tenemos el codito necesario para escribir un valor en el registro de desplazamiento, la nueva rutina/función solo deberá encargarse de "llamar" 60 veces seguidas a la que vimos antes, con el valor "0".

Veamos como hacerlo en PIC BASIC:

```
borro_segundero:
      bit_aux = 0  'Asigno el valor a enviar a
la variable auxiliar...
      For i = 0 To 59  '"i" irá de 0 a 59, de 1 en 1.
        Gosub escribo_segundo  'envío bit_aux
al registro
      Next i
Return
```

Código BASIC de la subrutina "borro_segundero".

Y ahora, lo mismo pero en CCS:

```
void borro_segundero(void){
   int i;
   for (i=0;i<60;i++) {  // "i" irá de 0 a 60,
de 1 en 1.
      escribo_segundo(0); //Envio un "0" al
registro de desplazamiento
   }
```

Código CCS de la función "borro_segundero".

Esto es todo lo que necesitamos saber para manejar correctamente los 60 LEDs del relojito. ■

www.todopic.com.ar

//Ariel Palazzesi//
arielpalazzesi@gmail.com

PIC BASIC

capítulo II

En la primera entrega aprendimos los rudimentos de la programación en BASIC. Ahora veremos cómo utilizar los pines del microcontrolador como entradas, permitiéndole conocer que está ocurriendo en el mundo real.

Como vimos antes, los pines del microcontrolador pueden configurarse como entradas o salidas. Sabemos que en caso de utilizarlos como salidas pueden emplearse para controlar el encendido de un LED o, mediante la interfaz adecuada, de prácticamente lo que se nos ocurra. Pero eso solo representa la mitad del poder de un microcontrolador. El otro 50% de su fortaleza reside en su capacidad de poder recibir estímulos externos. Para ello, debemos aprender a leer los estados lógicos presentes en los pines configurados como entradas.

.Leyendo el estado de un pulsador

Seguramente no hace falta que te explique que es un pulsador, pero como nos interesa que todo quede perfectamente claro, lo vamos hacer de todos modos: un pulsador es un dispositivo que al presionarlo permite que la corriente lo atraviese, cerrando el circuito.

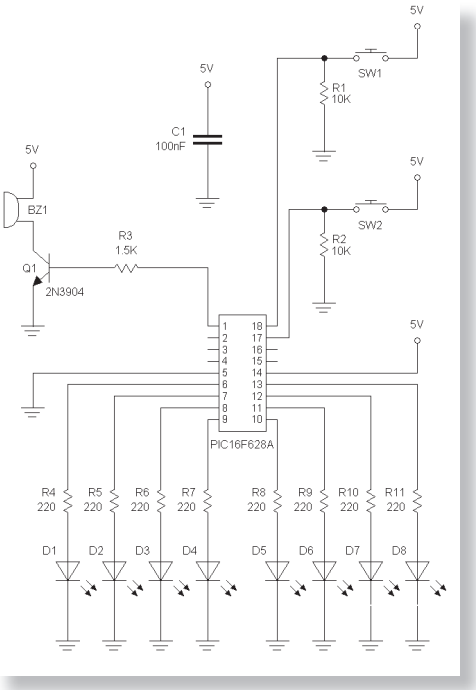


Figura 1: Este es el esquema que utilizamos para nuestros ejemplos

En el esquema que vimos en la anterior entrega, y que ponemos otra vez aquí por si te lo perdiste, puedes ver que hay dos pulsadores. Uno conectado al pin 17 (PORTA.0) y otro al pin 18 (PORTA.1). El otro extremo de cada pulsador está conectado a 5V, por lo que al presionar cualquiera de los pulsadores, el pin correspondiente del micro se pondrá en estado lógico alto (es decir, a "1").

Dado que eres muy observador, seguramente has notado que en cada pulsador también hay conectado un resistor que tiene un valor de 10K. La función que tienen esos componentes es mantener el pin correspondiente del micro "anclado" a 0V cuando el pulsador está abierto. Si no las pusiéramos, cualquier ruido eléctrico podría ser erróneamente interpretado como un accionamiento del pulsador. Si temes que al pulsar SW1 o SW2 se provoque un cortocircuito entre los 5V y GND, quédate tranquilo: justamente para eso están los resistores. Su gran valor (comparado con la baja resistencia interna de las entradas del PIC) garantiza que solo una muy pequeña corriente circulara a través de él, por lo que no hay problemas en que estén ahí. Confía en nosotros.

Tan común es la práctica de poner un resistor en esa posición, que hasta tiene un nombre: resistor pull-down. Si en lugar de estar conectado a GND lo estuviese a +V, recibiría el nombre de pull-up.

Pero volvamos a nuestro tutorial de BASIC. El siguiente código se encarga de configurar todos los pines del micro como E/S, deshabilitando los comparadores. Luego, pone todo el puerto A como entradas y el B como salidas. No vamos a usar más que un pin de cada uno, pero igualmente por ahora pondremos cada puerto completo en el mismo estado.

Una vez hecho esto, hay un bucle comprendido entre las líneas **LOOP**: y **Goto loop**, que se encarga de repetir infinitas veces la línea que está entre ellas: **PORTB.0 = PORTA.0.**

```
AllDigital
 TRISA = %11111111
 TRISB = %00000000
 loop:
    PORTB.0 = PORTA.0
 Goto loop
```

Justamente es esa la línea principal dentro de nuestro programa. Como ya habrás deducido, la instrucción **PORTB.0 = PORTA.0** hace que el valor del bit 0 del **PORTB** tome el valor del bit 0 del **PORTA.** Que ambos bits sean el cero es solo una coincidencia, se podrían haber elegido otros valores.

Al ejecutarse el programa, cada vez que se accione el pulsador conectado a **PORTA.0** (SW2, en el diagrama), ese pin se pondrá a estado alto, ya que la corriente circulara desde +V al pin 17 del PIC a través suyo. Ese "estado alto" se interpreta dentro del PIC como un "1", y es el valor que se le asigna a PORTB.0, con lo que el también pasara a estado alto. Eso provocara que el LED conectado en ese pin se ilumine.

Cuando soltamos el pulsador, **PORTA.0** vuelve a estado bajo, ya que se pone a masa a través del resistor de 10K (R2 en el diagrama), y **PORTB.0** hará lo propio, apagando el LED. Todo lo que hace nuestro sencillo programa es "copiar" en el LED el estado del pulsador.

Vamos a simularlo en el **PIC SIMULATOR IDE**, que para eso lo hemos comprado. Lo primero es abrir la ventana del compilador **BASIC**, y copiar en ella el código de más arriba. Debería quedar más o menos como vemos en la figura 2. Luego, lo compilamos y cargamos en el emulador presionando la tecla F9.

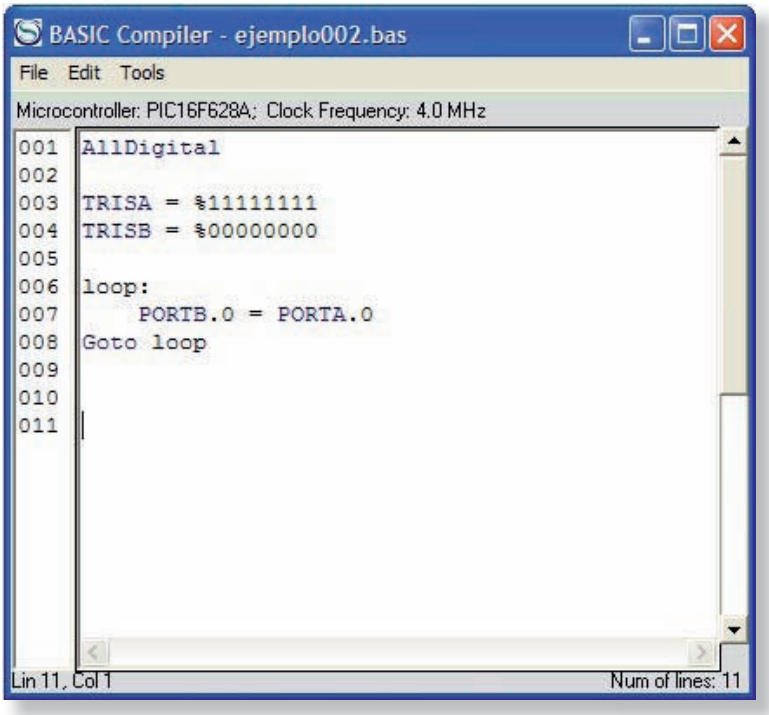


Figura 2: Este es el esquema que utilizamos para nuestros ejemplos

PIC SIMULATOR IDE transformará nuestro código **BASIC** en un archivo **HEX** listo para grabar en un **PIC** real o para ser simulado. Aparecerá el cuadro de dialogo que nos informa que no han ocurrido errores y que el tamaño del programa es de 20 words.

Si volvemos a la ventana principal del PIC SIMULATOR IDE, y desde "Tools" -> "Microcontroller View" abrimos la vista del microcontrolador, al darle "Start" a la simulación tendremos algo parecido a lo que muestra la figura 3: el pin 6, correspondiente a **RB0** está en "OFF" porque el pulsador del pin 17 **(RA0)** está en OFF. Si con el mouse hacemos un clic sobre la "T" que está al lado del pin 17, la vista del microcontrolador pasará al estado que muestra la figura 4.

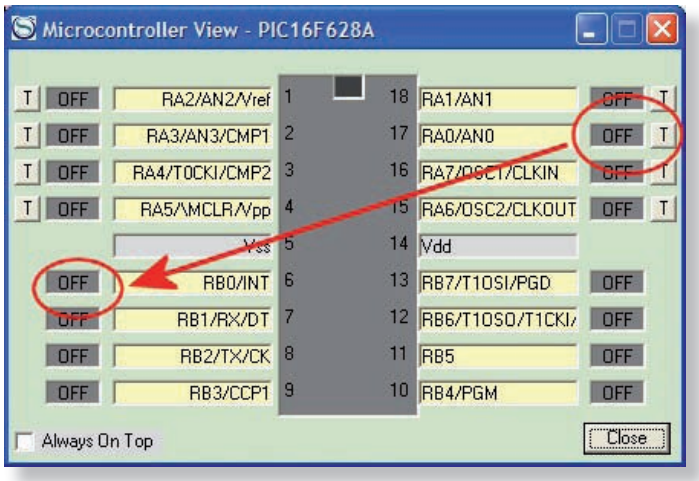


Figura 3: Pulsador abierto, LED apagado.

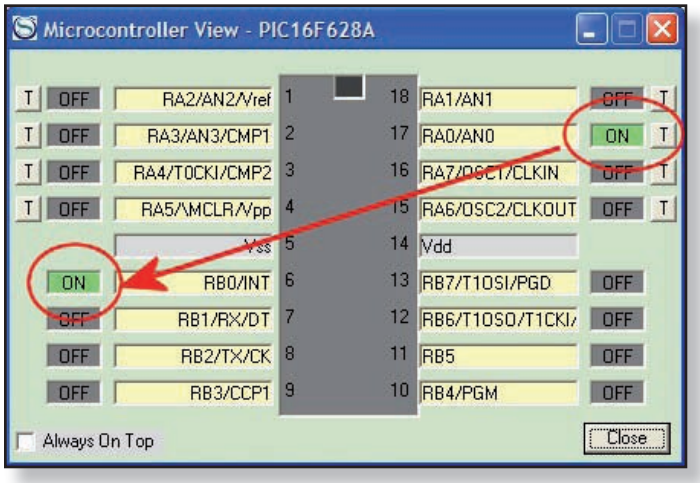


Figura 4: Pulsador cerrado, LED encendido.

Recordemos que el botón "T" significa "cambio" (Toggle) por lo que el estado del pin 17 permanecerá en alto hasta que lo pulsemos otra vez, y el estado del microcontrolador volverá a ser el inicial. Como en cualquier curso, conviene realizar estas prácticas, que aunque puedan parecer muy sencillas nos ayudaran a conocer las herramientas disponibles y "tomar confianza" al programa. También es interesante el realizar cambios en el programa BASIC, recompilar y analizar los resultados.

En la próxima entrega comenzaremos a ver las estructuras de control de flujo del programa que dispone PIC BASIC. Estas nos permitirán encarar programas más útiles y complejos. ■

µCONTROL
Electrónica en General Pics en Particular
www.ucontrol.com.ar

//Mario Sacco//
service.servisystem@gmail.com

hablemos de antenas

Cuando pensamos en controlar un dispositivo a distancia a través de RF, nos encontramos con la difícil elección de la antena apropiada y la posterior construcción de la misma. Veamos cómo develar ésta ciencia oculta.

Cada vez que intentamos controlar un dispositivo de forma remota, mediante ondas de radio, buscamos cumplir con la ley fundamental que rige las comunicaciones: "Llegar con nuestra señal, lo más lejos posible".

Muchos de éstos intentos de maximizar la distancia útil y efectiva de nuestro control, han finalizado con resultados nefastos. Esto es especialmente cierto en el hobby de los aviones radiocontrolados, por mencionar algún ejemplo, ya que no siempre el dispositivo remoto se quedará quieto ó se detendrá sin ocasionar daños así mismo ó a terceros. Muy por el contrario, pueden llegar a actuar de forma tan impredecible, que nos arruinaría el trabajo e ilusión de meses, en un instante.

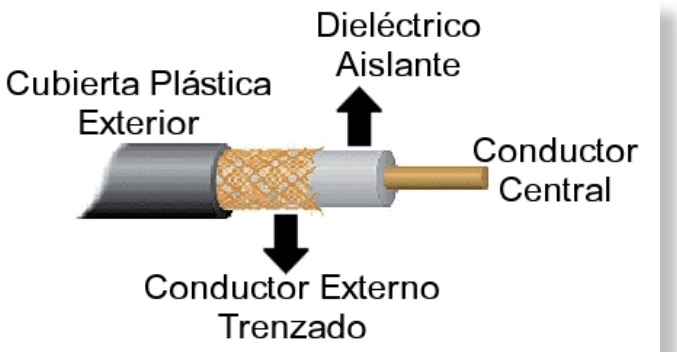
Por lo tanto será muy útil aprender las sencillas y elementales técnicas de construcción de antenas, para así poder manejar los robots ó sistemas inteligentes que construyamos, mucho más allá de lo que nuestro elemental juego de módulos de RF nos permitan por sí solos.

A medida que vayamos leyendo sobre el tema, iremos aclarando ciertos conceptos y nombres técnicamente correctos, que usaremos a lo largo de las explicaciones para no sólo aprender de antenas, sino también referirnos a ellas y su entorno de forma apropiada.

Podemos definir que un Receptor (en adelante Rx), podrá recibir señales desde un Transmisor (en adelante Tx), el que estará apropiadamente conectado, mediante una Línea de Transmisión, a una Antena. Esta deberá ser instalada a la mayor altura posible.

Lógicamente si es necesario, nuestro Rx también estará conectado a una Antena para optimizar los alcances del enlace radioeléctrico.

Tal cómo habíamos mencionado, el medio que unirá nuestro equipo (sea Rx ó Tx) con la Antena, será una Línea de Transmisión, que por su simpleza de uso y fácil adquisición, construiremos utilizando Cable Coaxil o Coaxial.



Aspecto de un Cable Coaxil

El conductor central y la malla trenzada exterior serán el nexo con nuestro equipo.

> **"El coaxil utilizado en la realización de la antena posee una atenuación a las señales de radio que es proporcional a la longitud del mismo"**

Entre los cables más populares que encontramos en el mercado están el RG-58, de 50 Ohms de impedancia característica, y el RG-59, de 75 Ohms. Este último se ha popularizado gracias a su uso en instalaciones de TV. El anterior, en cambio, se lo conoce más por su amplio uso en el ámbito de las telecomunicaciones y es del cuál encontraremos mayor variedad de calidades constructivas.

Éste es un punto muy importante a tener en cuenta al momento de seleccionar el modelo de cable a utilizar, ya que los coaxiles poseen una característica nociva para nuestros propósitos, que es la de atenuar las señales de radiofrecuencia que lo atraviesen. Esta atenuación es directamente proporcional a la frecuencia de trabajo y a la longitud del cable mismo. Dicho en palabras simples: A mayor frecuencia, mayor atenuación.

Las frecuencias libres de uso asignadas dentro del Espectro Radioeléctrico, y por lo tanto ideales para nuestros propósitos, son las consideradas "altas" y se encuentran en la banda conocida cómo UHF y Microondas. Estas frecuencias son 305 MHz., 418 MHz., 433 MHz. y 2,4 GHz., por mencionar algunas.

Existen otras, en otras bandas, cómo 27 MHz. Y 72 MHz., pero se encuentran en desuso en la actualidad, en gran parte gracias a que los pequeños módulos de UHF, han venido a resolver muchos de los problemas y

limitaciones que se planteaban al momento de pensar en un enlace vía radio.

Un cable RG-59 de buena calidad, nos proporcionará menor atenuación que un cable RG-58 común, además de ser más económico, por lo que para nuestros fines y mientras manejemos bajas potencias (valores menores a 1 Watt), será una elección acertada, obteniendo un buen equilibrio Precio/Performance.

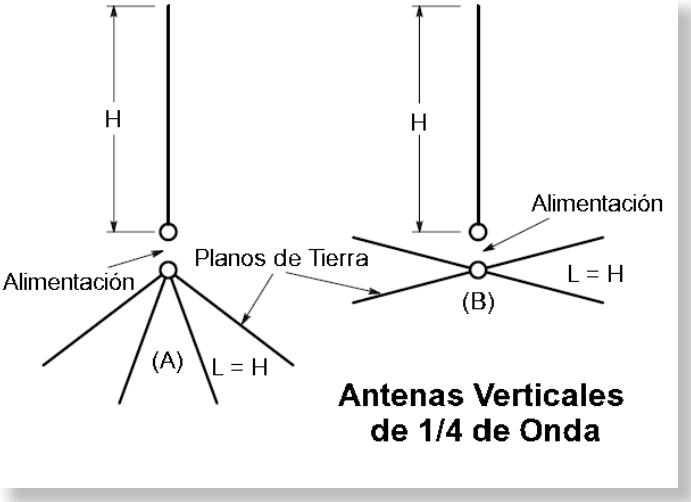
.Ahora sí, Las Antenas

Existen tres modelos prácticos de antenas, realizables de forma sencilla en no más de un par de horas, que enumerarlas de la siguiente forma:

- *Vertical de ¼ de Onda*
- *Dipolo de ½ Onda*
- *Direccional Yagi*

La más sencilla de construir, y por lo tanto la primera que desarrollaremos, es la Antena Vertical de ¼ de Onda, también conocida cómo Ground-Plane o con el simpático nombre de "paragüitas", en alusión al formato que le dan los planos de tierra.

En su nombre encontramos resumida la arquitectura constructiva de ésta antena: un Elemento Irradiante, en posición vertical, conectado al conductor central de la Línea de Transmisión, y un segundo elemento conformado por un "Plano de Tierra", donde irá conectada la malla



Vista teórica de la misma antena vertical con dos planos de tierra de diferentes formas.

El análisis de las imágenes de la figura anterior nos revela muchos aspectos de ésta sencilla, pero altamente efectiva, antena.

Una primera observación nos muestra que, por la posición vertical del Elemento Irradiante, la radiación de energía se dará en forma Omnidireccional, es decir, nuestra señal transmitida saldrá de la antena y viajará en todas las direcciones por igual, cubriendo los 360º a su alrededor.

La segunda apreciación nos dice que la señal estará Polarizada en forma vertical, lo que significa que la otra antena que se enlazará con ésta deberá poseer la misma Polarización, ya que si no coincide con ella, la señal sufrirá una importante atenuación.

Un Elemento Irradiante montado en forma paralela al suelo trabaja polarizado en forma horizontal, mientras que si está montado en forma vertical trabaja, como es obvio, polarizado en sentido vertical.

Este concepto parece superfluo ahora, paro tendrá una gran importancia cuando realicemos un enlace Full-Duplex.

Otro detalle no menor es la posición que otorguemos respecto a la vertical al plano de tierra de nuestra antena. Distintos ángulos de inclinación de este plano nos proveerán distintos valores de Impedancia de la antena.

"ROE (ó SWR, por "Standing Wave Ratio" en Inglés), es la relación que existe entre las Potencias Incidente y Reflejada"

Por simple deducción podemos entonces decir que cuanto más se aproximen entre sí los valores de impedancia de nuestra antena con el correspondiente al coaxil seleccionado, mejor acoplamiento obtendremos entre ambos dispositivos y mayor será la energía transmitida hacia la antena. Es decir, menores serán las desadaptaciones que impedirán la llegada de nuestra señal a la antena. Esto es muy importante que lo tengamos claro para cualquier tipo de antena que intentemos construir.

Las dimensiones de los Elementos que forman nuestra antena, la forma mecánica de su construcción y el medio circundante para el emplazamiento elegido, regirán el valor final obtenido de la Impedancia de nuestra antena. Una antena ubicada a ras del suelo y rodeada de edificios, no tendrá el mismo rendimiento que si la emplazáramos en una elevación libre de obstáculos y con un enlace visual con su compañera Rx / Tx

.El enemigo acecha y se llama R.O.E.

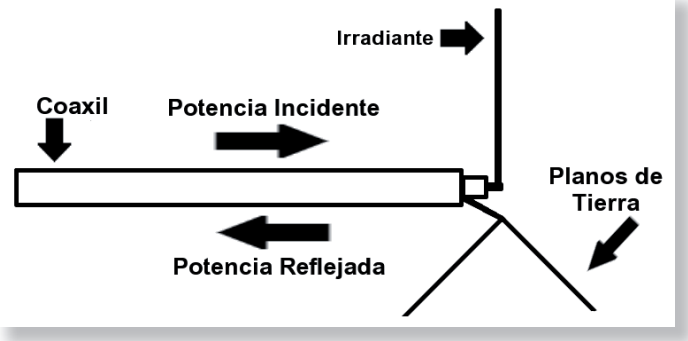
Una antena desadaptada respecto del cable coaxil que la enlaza al equipo, una mala construcción mecánica, las deficiencias provocadas por el paso del tiempo y el clima, el envejecimiento de los materiales y muchos otros factores serán los que al cabo del tiempo, irán deteriorando nuestro sistema de antena. Este hecho se representará como una variación de la impedancia característica del sistema conectado al equipo.

En estos casos, una parte proporcional de la energía que entregamos a nuestra antena para que ésta se encargue de irradiarla al espacio, será devuelta al transmisor, ya que nuestro sistema no irradiará al 100% debido a los posibles defectos mencionados.

Técnica hablando podemos decir que cuando una parte de la Potencia Incidente, que estamos enviando a

nuestra antena retorna al equipo como Potencia Reflejada, estamos siendo presas de lo que se conoce como Relación de Ondas Estacionarias, comúnmente llamada ROE.

ROE (ó SWR, por "Standing Wave Ratio" en Inglés), es la relación que existe entre las Potencias Incidente y Reflejada y su valor está regido por fórmulas matemáticas y conceptos mecánicos que sería innecesario explicar ahora, aunque sí es bueno saber de la existencia de éste fenómeno.



La energía que no puede ser irradiada al espacio retornará por el mismo coaxil hacia el Tx, pudiendo deteriorarlo.

Cuando los valores de ROE son altos corremos un serio riesgo de destruir la etapa de potencia de salida de nuestro transmisor, ya que el mismo está pensado y preparado para entregar energía y no para recibirla.

Comprendamos entonces cuán importante es leer muy bien la hoja de datos del fabricante de nuestro Tx y Rx, para conocer las Impedancias características que ellos nos recomiendan utilizar a la salida de los módulos, y así obtener el máximo beneficio de ellos.

.Manos a la Obra

Para construir nuestra antena debemos conocer las medidas apropiadas que debe tener la misma. Para el caso de la antena vertical de ¼ de onda, la formula es la siguiente:

72/Frecuencia (Mhz.) = H = L (Metros)

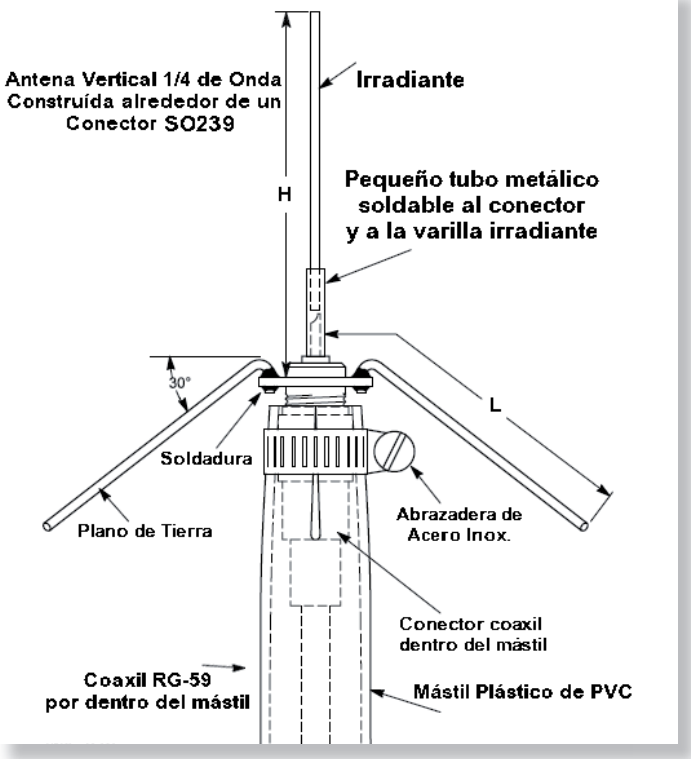
Conociendo ésta fórmula ya podemos saber las dimensiones de los elementos que compondrán la misma. Por ejemplo para un módulo que transmita ó recepcione en una frecuencia de 433,920 Mhz., la fórmula resultaría en lo siguiente:

72/433,920 Mhz. = 0,1659 Metros = 16,59 Centímetros

O lo que es lo mismo, aproximadamente 6.53 pulgadas.

Ahora es el momento de decidir con qué materiales construiremos la misma. Cualquier material metálico puede sernos útil para éste fin. Cobre, hierro, aluminio, zinc, bronce y/o cualquier otro que tenga propiedades conductivas y posea la rigidez mecánica suficiente cómo para entregarnos una construcción robusta, firme y duradera. Eso siempre quedará a criterio de cada uno de Uds. y lo realizarán con los elementos que posean al momento de la construcción.

Otra opción, es seleccionar con antelación los materiales y acopiar los mismos de a poco, a los efectos de realizarla con materiales seleccionados especialmente, cómo por ejemplo, buenos aluminios, buenos soportes, buenos mástiles, etc.



Un poco de aluminio, un juego de conectores apropiados, y nuestra antena estará lista.

Puede observarse en el gráfico que a partir de un conector UHF Hembra, conocido como SO239, podemos armar nuestra antena.

Los planos a tierra podrán ser soldados a los orificios que posee el conector hembra, que nos servirá de base constructiva, y doblados suavemente hacia abajo, dándoles un ángulo a los elementos (planos de tierra) de entre 30 grados (mínimos) hasta 60 grados (máximo).

Al terminal de conexión del conductor central le soldaremos el elemento irradiante (la varilla que quedará en posición vertical), ó se la adosaremos mecánicamente de la forma que creamos más conveniente, efectiva y firme.

La llegada de la señal a transmitir, ó el envío al receptor de la

Imagen del Conector SO239.

señal captada, se logra a través del coaxil que recorre internamente el mástil de plástico o cualquier otro material aislante, para terminar en un conector PL259 roscado con justeza a nuestra base original de construcción que es la hembra SO239.

Una sencilla abrazadera de material inoxidable, concluye la construcción.

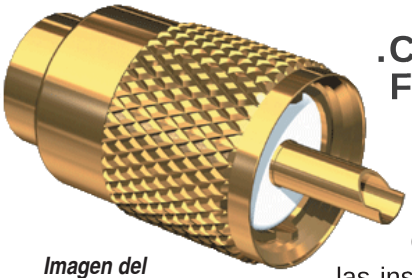


Imagen del Conector PL259.

.Consideraciones Finales

Una vez finalizada la construcción, si hemos comprendido y seguido las instrucciones vertidas en éstas páginas, estaremos listos para conectar nuestra/s antena/s a los módulos que estarán instalados en nuestros sistemas microcontrolador. Lo primero que haremos es tratar de dotar a la instalación de nuestro sistema irradiante (recordemos que se llama de igual forma, sea para Tx, cómo para Rx), de la mayor altura posible y alejada de obstáculos circundantes que impidan la correcta emisión en algún sentido y que puedan afectar la impedancia del sistema provocándonos ROE.

Un aspecto importante a recordar es que el coaxil utilizado en la realización posee una determinada atenuación a las señales de radio, la que es proporcional a la longitud del mismo, por lo que para las frecuencias de UHF y para los módulos comerciales de algunos pocos miliWatts de potencia, no podremos abusar de la cantidad de metros que utilicemos.

Una medida de aproximadamente entre 5 y 10 metros nos dará una buena relación entre longitud/atenuación de señal y los resultados obtenidos serán más que sorprendentes.Siempre querremos llegar más allá.

Sólo nos falta darnos cuenta, dónde y cómo utilizaremos ésta antena simple pero que nos hará ganar muchos metros en nuestro enlace de mando. Si con los módulos originales, sin antena, o con algún trozo de cable sencillo lográbamos varias decenas de metros, tengan por seguro que con antenas exteriores, esta distancia puede llegar a ser de varias centenas o más.

Esto significa nuevos horizontes en la aplicación y uso de módulos de UHF, en sistemas enlazados vía radio.

No se pierdan el próximo número de µControl, donde veremos cómo realizar antenas direccionales de ganancias muy altas, para llegar mucho más lejos aún. Que la disfruten.

Referencias:

Antenas Yagui: http://es.wikipedia.org/wiki/Antena_Yagi
ROE: http://www.solred.com.ar/lu6etj/tecnicos/roe/roe.htm

//Manuel Jiménez Franco//
nocturno@micropic.es

receptor para el protocolo DMX512

El protocolo DMX512 es el estándar por excelencia para el intercambio de información entre el equipamiento de tecnologías teatrales. De diseño sencillo, fácil uso y gran difusión en el mercado de tecnología para el entretenimiento, DMX512 llegó para quedarse en cualquier tipo de escenario moderno. En el presente artículo aprenderemos un poco acerca de este protocolo y como construir y utilizar un receptor-visualizador DMX512.

.El protocolo

El protocolo DMX512 nació en 1986 en la USITT (Instituto americano de tecnologías teatrales) para convertir el sistema de comunicación entre consolas y dimmers en un estándar eficiente. Ha sufrido diversas revisiones hasta evolucionar al estándar actual, conocido oficialmente como "Entertainment Technology — USITT DMX512–A — Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories", que fue aprobado por ANSI en noviembre de 2004. El actual estándar es también conocido como "E1.11, USITT DMX512–A", o solo "DMX512-A", y es mantenido por la ESTA (The Entertainment Services & Technology Association).

DMX fue originalmente pensado para controladores de enlace y dimmers de diferentes fabricantes. La sencillez de su diseño y la licencia GNU, han provocado que en la actualidad sea un estándar, de amplia difusión en el mercado, para el control de todo tipo de dispositivos de iluminación, scanners, máquinas de humo, cabezas móviles. Prácticamente todos los dispositivos utilizados en espectáculos incorporan DMX.

Advertencia: Como DMX512 es un sistema de transmisión de datos sin mecanismos para el control de errores, no debe ser usado para controlar pirotecnia.

.Nivel Físico

El DMX512 es un protocolo de transmisión de datos serie que se basa en el estándar internacional EIA – 485 o RS485, éste se emplea en aquellas aplicaciones donde se necesita una transmisión serie fiable y simple. A diferencia del conocido RS232, RS485 permite cubrir distancias superiores y tiene una serie de ventajas que lo hacen superior a RS232 para muchas aplicaciones donde la distancia y fiabilidad de la comunicación son requisitos importantes.

La información se transmite por una pareja de conductores y no mediante un solo conductor. Este tipo de canales de transmisión se conoce como: líneas balanceadas o diferenciales; su característica es la elevada inmunidad a los ruidos eléctricos y electromagnéticos comunes (referidos a masa).

Como las señales que nos interesan son las relativas a la información de los canales DMX, estas señales son trasmitidas, ex profeso, de modo diferencial y posteriormente son amplificadas sin aumentar el ruido que se presenta, generalmente, de modo común (con la misma polaridad respecto a masa). Los amplificadores diferenciales utilizados actualmente en RS485 son en realidad pequeños circuitos integrados llamados **line driver** y **line river**. El primero es el emisor que está instalado en la consola y el segundo es el receptor que estará instalado en cada uno de los dispositivos a controlar.

.Nivel Lógico

El protocolo DMX512 se basa en la utilización de "canales" para transmitir comandos a los dispositivos receptores. DMX512 tiene un límite de 512 canales por universo (DMX universe), y cada canal ocupa un byte. Por lo que el valor transmitido por canal se puede regular desde 0 hasta 255: son los "valores" DMX (DMX values). Cada trama DMX512 lleva los 512 bytes correspondientes al dato de cada canal, independientemente de que se utilicen o no todos los canales. Las mesas profesionales que usan DMX pueden soportar hasta 8 universos DMX y con la tecnología EtherDMX estos pueden ser ampliados aún más.

En DMX512 se transmiten los datos de modo asincrónico, a 250Kbit por segundo. Esto significa que las

señales del transmisor y de los receptores no están en sincronía, pero los receptores se sincronizan al mando de la consola cada vez que ésta envía una determinada señal.

Básicamente, y aunque luego lo veremos en profundidad, la trama DMX completa se compone de una señal de sincronización y a continuación los 512 bytes de información que se corresponden con los 512 valores DMX. Los receptores reciben toda la trama, pero procesan sólo la información relativa a los canales para los que están configurados.

La información se transmite siguiendo este diagrama de tiempos:

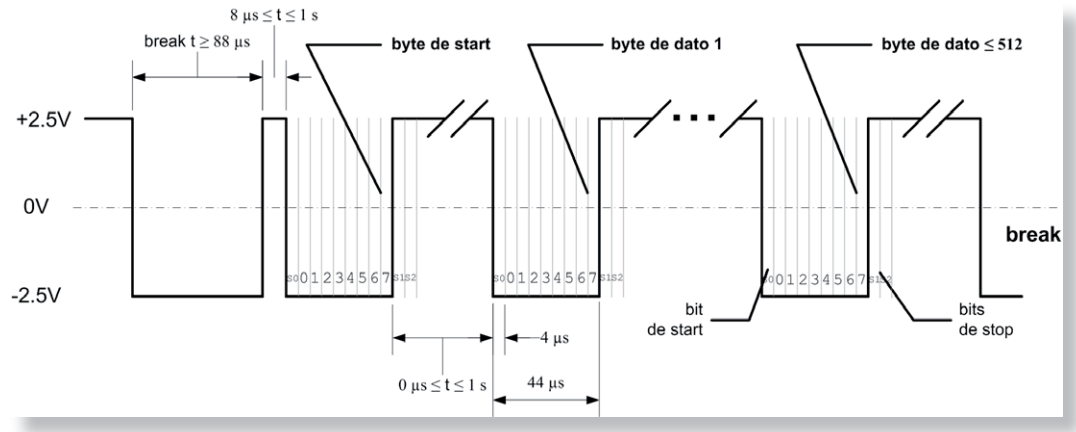


Diagrama de señales y tiempos del protocolo

Como se observa, la trama completa tiene varias partes:

- ***señal BREAK:*** es un nivel bajo con un mínimo de 88us
- ***marca tras BREAK:*** nivel alto con un mínimo de 8us
- ***byte Start:*** el byte Start siempre vale 0
- ***tiempo entre bytes:*** es un nivel alto que puede llegar hasta 1 s
- ***trama de 512 bytes:*** aquí aparecen los datos de los 512 canales

Cada byte se transmite con:

- ***un bit de start*** a nivel bajo
- los ***8 bits de datos***
- ***dos bits de stop*** a nivel alto

De esta manera, algunas consideraciones de tiempo respecto al protocolo son las siguientes:

- duración mínima para una trama completa: ***22,7ms***
- máxima velocidad de refresco de la información: ***44 veces por segundo***

.Instalaciones DMX

Topología

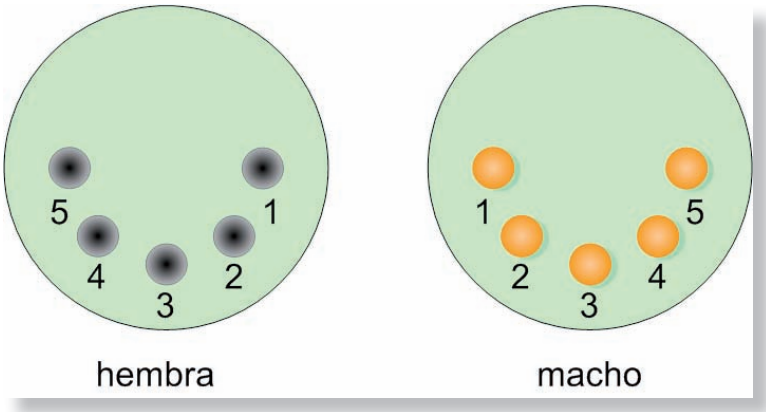
Con DMX512 se pueden alcanzar distancias de comunicación, entre los extremos del BUS, de hasta 500m. Sin embargo esta distancia es para un caso más o menos ideal, pero en la práctica es raro encontrar líneas de transmisión con más de 100m de longitud. Si fuese necesario aumentar las distancias es preciso utilizar buffers o splitters.

Habitualmente, los dispositivos DMX tienen una entrada y una salida, de manera que es fácil configurarlos en cadena, uno tras otro. Es obligatorio colocar en los extremos del BUS una resistencia de terminación de 120ohm 1/4W, entre los conductores de las señales de dato. Olvidar colocar esta resistencia suele ser causa frecuente de averías en instalaciones DMX o fallas en la comunicación.

.Cables y conectores

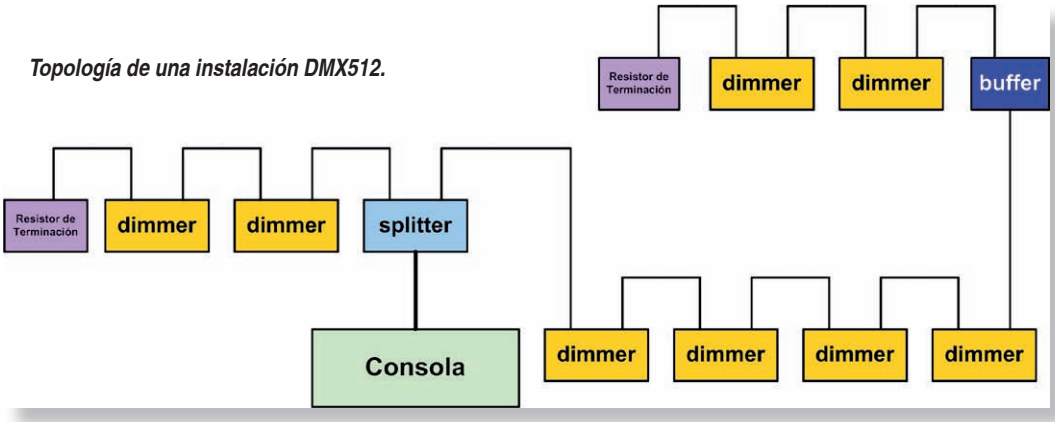
Los cables habitualmente utilizados en DMX son los de par trenzado, dado que ofrecen una mayor inmunidad al ruido. Como hemos comentado antes, ambos hilos reciben la misma cantidad de ruido y ello permite a los amplificadores diferenciales eliminarlo para quedarse con la información válida.

Los conectores estándar son los XLR, de los que hay dos modelos: con 3 o con 5 pines. Antiguamente se utilizaban mucho los de 3, pero en la actualidad son más frecuentes los de 5 pines.



Conectores XLR de 5 pines. Éstos son los más utilizados en las instalaciones DMX512.

Topología de una instalación DMX512.



La configuración de los pines 1 al 3 en un cable de 3 pines es la misma a la de los pines 1 al 3 en un cable de 5 pines.

Un cable de 5 pines está configurado de la siguiente forma:

Pin 1 = señal de referencia = revestimiento del cable (malla o pantalla);

Pin 2 = señal invertida = "-" polo negativo;

Pin 3 = señal = "+" polo positivo;

Pin 4 = opcional (la utilización de este pin varía de acuerdo con el aparato en operación y los fabricantes nunca llegaron a un acuerdo sobre cómo utilizarlo. La intención original era tener realimentación de los aparatos y establecer son estos un enlace bidireccional).

Pin 5 = opcional (la utilización de este pin varía de acuerdo con el aparato en operación y los fabricantes nunca llegaron a un acuerdo sobre cómo utilizarlo. La intención original era tener realimentación de los aparatos y establecer son estos un enlace bidireccional).

Si bien es cierto que en la industria del entretenimiento el cable del tipo XLR es el cable estándar, en algunas aplicaciones por motivos prácticos y estéticos es muy común la utilizacíón de cables del tipo UTP Cat5e (cable de red).

El armazón de los dispositivos nunca se conecta al pin 1, la señal de referencia, ya que podría provocarse un anillo de masas y afectar el correcto funcionamiento del sistema. Sin entrar al tema de los fenómenos causados por una equivocada conexión a tierra, de la funda defensiva del cable de transmisión de datos, es importante decir que una eventual conexión a tierra de este conductor se puede realizar en un sólo punto de todo el sistema y que normalmente este procedimiento se realiza sólo en las instalaciones fijas.

De todas maneras, es una buena práctica que los equipos dispongan en su entrada DMX de optoacopladores que eliminen la posibilidad de contacto eléctrico entre el bus DMX y la circuitería del dispositivo. La mayoría de los equipos profesionales disponen de aislamiento optoacoplado.

El ReceptorDMX que presentamos es de carácter puramente experimental, y sólo sirve para probar a interpretar el protocolo DMX512

.Dirección DMX

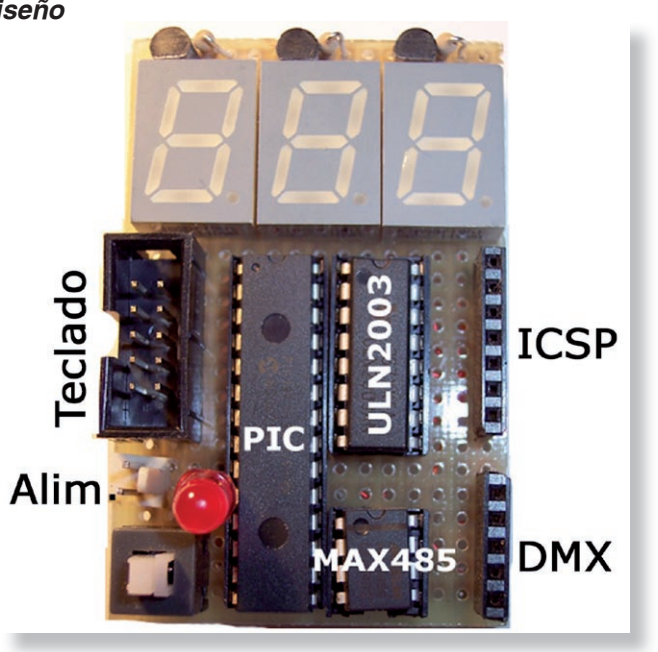
Puesto que en el bus DMX viaja la información de 512 canales, cada dispositivo debe ser configurado para "escuchar" los que necesite. Cuando un dispositivo necesita la utilización de varios canales, suele configurarse sólo la dirección del primero de ellos, quedando reservados para dicho dispositivo todos los que necesite a partir del primero.

Lo habitual es que un dispositivo utilice varios canales, y se identifique físicamente con el código del canal de inicio. De esta manera, si un cabezal proyector utiliza, por ejemplo, 16 canales, y está identificado con el código 128, automáticamente reservará los canales 128 a 143 para sí mismo. Lo único que hay que tener en cuenta es que las direcciones DMX de los aparatos nunca deben estar superpuestas.

.Construyamos un receptor DMX

El ReceptorDMX que presentamos es de carácter puramente experimental, y sólo sirve para probar a interpretar el protocolo DMX512 y mostrar en su display el valor del canal seleccionado en cada momento.

Diseño



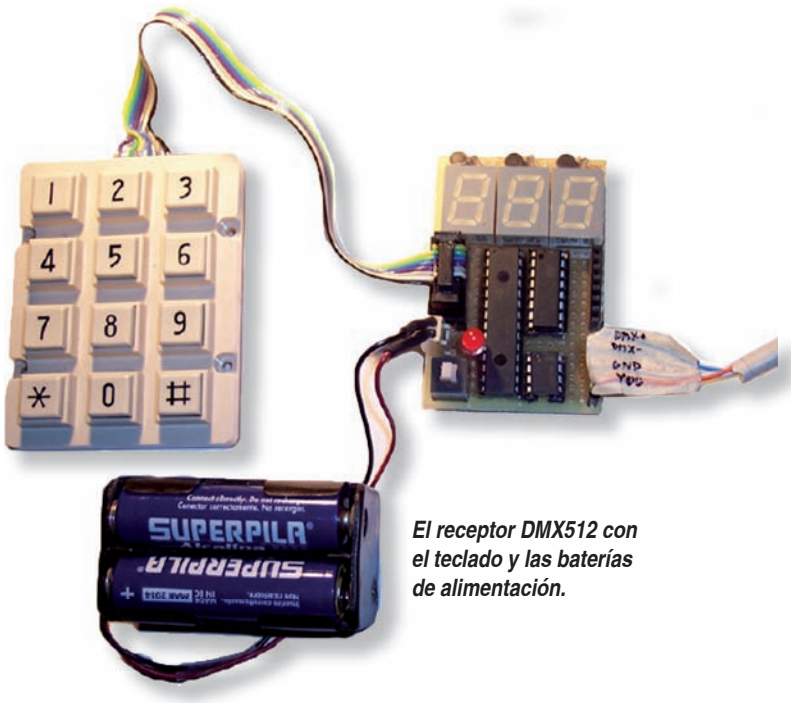
Nuestro receptor DMX512.

El circuito está compuesto por un PIC, como corazón, un arreglo de transistores Darlington ULN2003 para las lámparas de siete segmentos y un driver MAX485 para el bus DMX. Se ha utilizado un PIC 18F2550, pero el software podría compilarse para uno mucho más pequeño porque requiere muy poca memoria de programa.

El PIC almacena en su RAM los 512 valores de todos los canales DMX, lo que le permite "absorber" la información de la trama completa. El canal DMX se selecciona mediante el teclado matricial.

Hay otro dispositivo de salida: el LED. Ese LED está gestionado por PWM y su ciclo útil viene dado por el dato DMX del canal seleccionado. En este caso, no hay esperas: la información se muestra en tiempo real a la vez que se está escribiendo el canal seleccionado.

El conector DMX sólo necesita tres señales: DMX+, DMX- y GND. Se ha añadido Vdd porque permite alimentar algún circuito auxiliar que se pueda conectar al bus, aunque no es imprescindible.



El receptor DMX512 con el teclado y las baterías de alimentación.

Uso

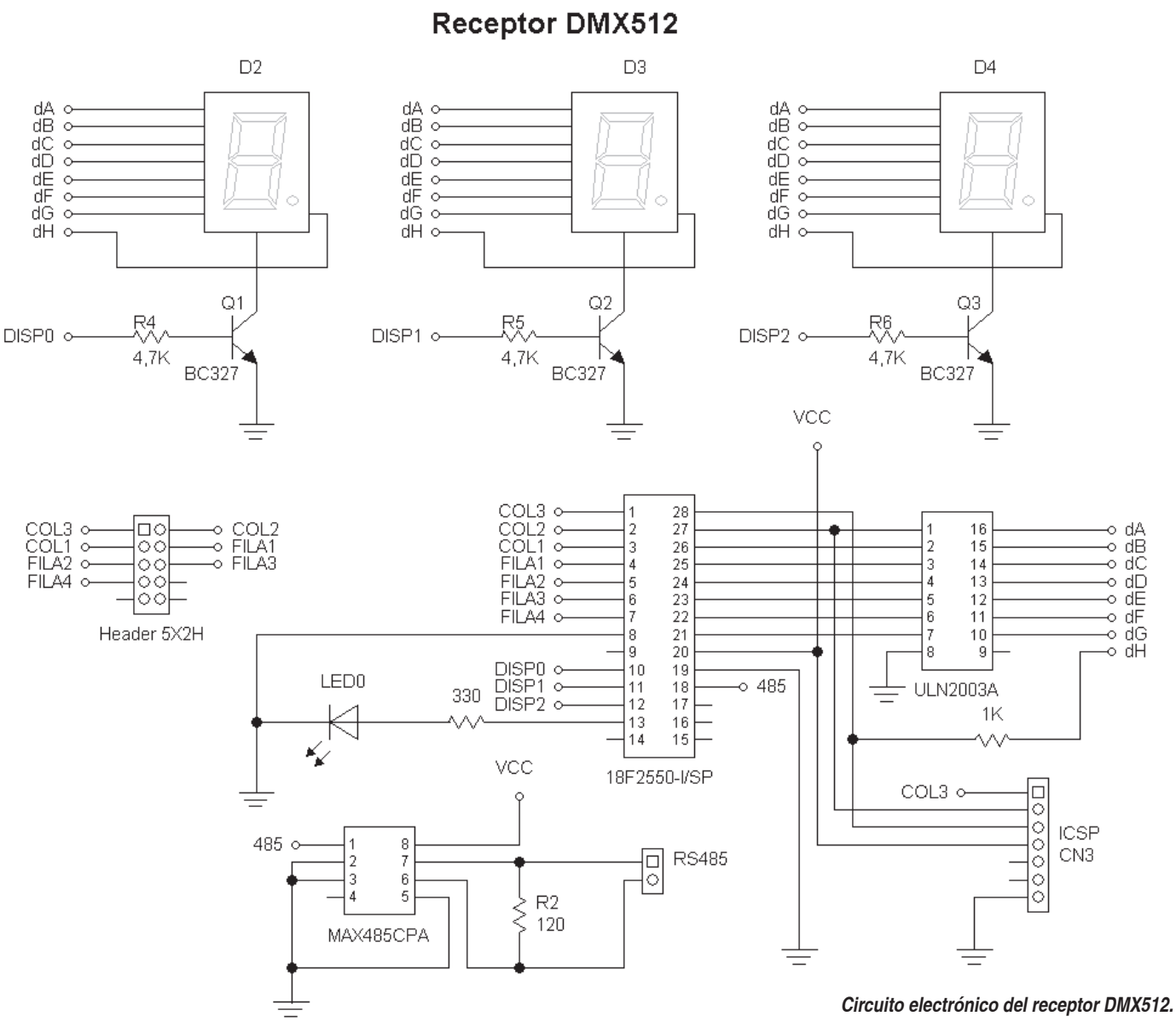
Utilizar el dispositivo es muy fácil. Al encender el dispositivo este comienza a capturar todas las tramas que le lleguen por el conector DMX. En el display aparece la palabra “OFF”, hasta que el usuario seleccione un canal con el teclado matricial.

A medida que se va pulsando el teclado, el display va mostrando el canal seleccionado. Mientras se toca el teclado, parpadean los puntos decimales de los displays para advertir que estamos viendo el selector de canal, en lugar del dato DMX. Al cabo de unos segundos la información aparece en los displays. El valor del canal seleccionado también se representa en el led, cuya intensidad es regulada mediante PWM.

Puesto que el dispositivo memoriza toda la trama (512 bytes) puede ser desconectado del bus y examinado a posteriori como si estuviera físicamente conectado.

Construcción

La construcción del receptor se basa en este esquema:



Circuito electrónico del receptor DMX512.

Del código fuente, sólo veremos aquí la rutina de atención a la interrupción de la USART, que es donde está el meollo de la cuestión.

El programa completo está disponible para descargar; las explicaciones del código están en los comentarios.

```
/**********************************************************************
*  Interrupción RDA: dato recibido por la USART                       *
*  Esta interrupción se activa cada vez que se recibe un dato en la   *
*  USART. Mediante el control de una máquina de estados se determina  *
*  la validez y el significado del dato recibido, y se obra en        *
*  consecuencia.                                                      *
*                        Nocturno – 2008              www.micropic.es *
**********************************************************************/
#int_rda
void Dato_Recibido_USART(void)
{
 while (RCIF) // ejecutamos mientras haya un dato pendiente de procesar
 {
  // Hacemos una copia del registro RCSTA porque sus bits cambian de valor
  // al leer RCREG y modificar CREN
  Copia_RCSTA.registro = RCSTA;
  // En RCREG está el dato que acaba de recibir la USART
  DatoRX = RCREG;
  // Si se reciben más de 3 bytes sin haberlos procesado, se produce un error
  // de Overrun. En este caso, se borra el error reiniciando CREN y dejamos
  // la interrupción preparada para procesar la siguiente trama DMX
  if (Copia_RCSTA.bits.OERR)
  {
    CREN=0;
    CREN=1;
    DMX_Estado = DMX_ESPERA_BYTE;
    return;
  }
  // Máquina de estados
  switch (DMX_Estado)
  {
    case DMX_ESPERA_BYTE: // si estamos en este estado y hay error FRAME
  // es que nos ha pillado en medio de un Byte. Hay que seguir esperando
  // hasta que desaparezca el error.
      if (!Copia_RCSTA.bits.FERR)
      // Ha llegado un byte. Ahora esperaremos la señal Break
        DMX_Estado = DMX_ESPERA_BREAK;
      break;

    case DMX_ESPERA_BREAK: // estamos esperando la señal Break
      // Esta señal se identifica porque aparece el error de Frame
      if (Copia_RCSTA.bits.FERR)
        // Tras recibir el error de Break, hay que esperar un byte de valor 0
        if (!DatoRX)
          DMX_Estado = DMX_ESPERA_START;
      break;
    case DMX_ESPERA_START: // ya hemos recibido el Break y ahora hay que
      // esperar un Byte con valor 0, que será la señal de Start
      // Mientras tanto, si recibimos un error de Frame, hay que volver a
      // empezar para recibir la señal de comienzo de trama.
      if (Copia_RCSTA.bits.FERR)
          DMX_Estado = DMX_ESPERA_BYTE;
      else {
        if (!DatoRX)
        {
          // Llegados a este punto, ya hemos recibido el Byte Start=0
          // y comenzamos la trama de valores DMX.
          DMX_Indice = 0;
          DMX_Estado = DMX_RECEPCION_DATOS;
        } else
```

```
            // Si el dato recibido no es 0, volvemos a empezar
            DMX_Estado = DMX_ESPERA_BREAK;
        }
        break;
      case DMX_RECEPCION_DATOS:
        // En este estado estamos recibiendo la trama de datos DMX
          // Si se detecta un error de Frame es que ha habido un error y estamos
          // al principio
        if (Copia_RCSTA.bits.FERR)
          if (!DatoRX)
            DMX_Estado = DMX_ESPERA_START;
          else
            DMX_Estado = DMX_ESPERA_BYTE;
        else
        {
          // Almacenamos el dato recibido en nuestro array
          TramaDMX[DMX_Indice++] = DatoRX;
          // Si ha llegado al final de la capacidad, cambiamos al estado de espera
            // de nueva trama
          if (DMX_Indice >= TotalCanales)
            DMX_Estado = DMX_ESPERA_BREAK;
        }
        break;
    }
}
return;
}
```

.Emisor DMX

Para comprobar que nuestro receptor funciona adecuadamente, es aconsejable contar con emisor DMX512. Si no se dispone de un dispositivo de prueba, es posible construir el receptor Manolator 256, cuyos esquemas y método de construcción se encuentra alojado en http://www.freedmx.com. También deben descargarse alguno de los programas de emulación de consolas DMX, que están alojados en FreeDMX.

.: servisystem :.
PRIMER PÁGINA ARGENTINA DEDICADA AL SERVICE DE TV , AUDIO , VIDEO Y A TODOS LOS AMANTES DE LA ELECTRÓNICA Y LOS MICROCONTROLADORES
INFORMACIÓN SOBRE:
REPRODUCTORES DE CD Y DVD // FIRMWARE PARA MP3 PLAYER
TUTORIALES DE TELEVISIÓN // TUTORIALES DE AUDIO
TELEVISIÓN DIGITAL // FOROS DE CONSULTAS
Y MUCHO MÁS...
www.servisystem.com.ar

uCONTROL
Electrónica en General Pics en Particular
Publicita aquí!!!
www.ucontrol.com.ar

//Ariel Palazzesi//
arielpalazzesi@gmail.com

monoestable con NE555

Existen algunos circuitos integrados que a pesar de permanecer durante años en el mercado, su gran utilidad hace que permanezcan vigentes, tal es el caso del temporizador NE555. Es por eso que lo hemos elegido para inaugurar esta nueva sección de la revista.

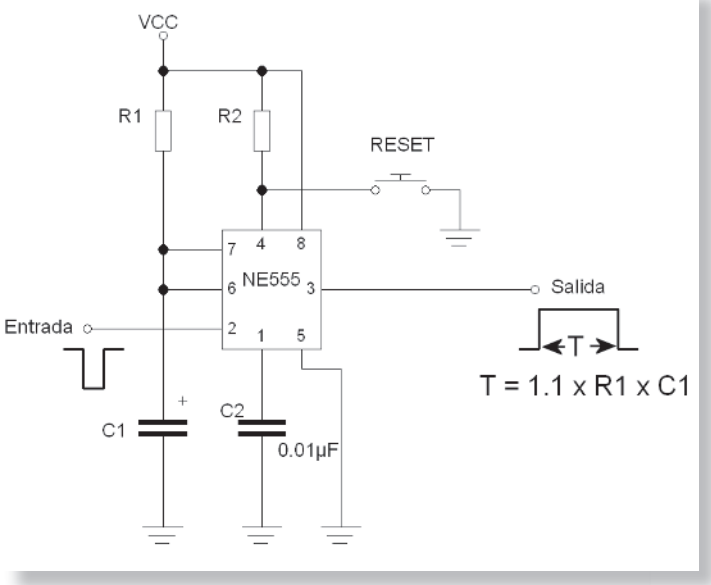
A pesar de ser un circuito integrado sumamente económico, se consiguen tiempos de temporización muy estables frente a variaciones de tensión de alimentación y de temperatura. La estabilidad en frecuencia es de 0,005% por °C. La precisión final estará dada por la calidad del condensador C1.

Un circuito monoestable recibe ese nombre por permanecer estable en un solo estado: el nivel bajo. En efecto, si conectamos el **NE555** de manera que se comporte como un monoestable (figura 1), su salida permanecerá en estado bajo, salvo en el momento en que reciba una señal en su pin numero 2, denominado **TRIGGER** (gatillo o disparador), en cuyo caso la salida pasara a nivel alto durante un tiempo T. Este tiempo está determinado por los valores del resistor **R1** y el condensador **C1**, de acuerdo a la formula de la figura 1. En ella, el periodo T se expresa en segundos, **R1** en Ohms y la capacidad de **C1** en Faradios.

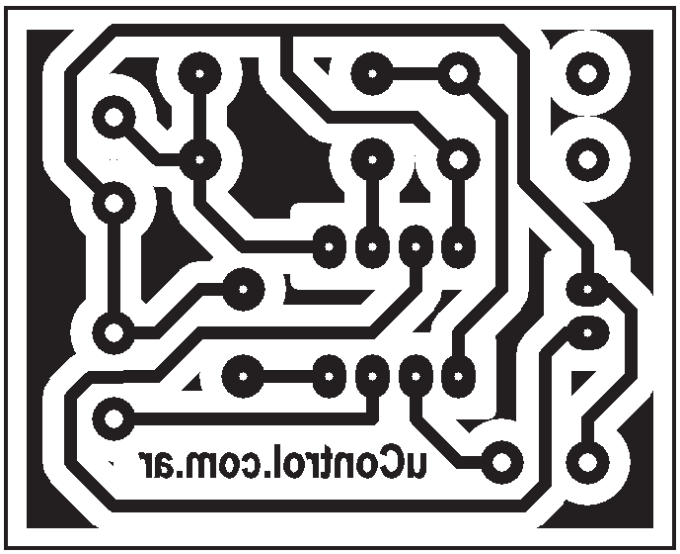
Un circuito monoestable recibe ese nombre por permanecer estable en un solo estado: el nivel bajo

Al presionar el pulsador identificado como **"TRIGGER"**, la salida (pin 3) del **NE555** pasará a estado alto y permanecerá así hasta que transcurra el tiempo fijado por el valor de **R1** y **C1** o hasta que se presione el pulsador **"RESET"** (lo que ocurra primero). En general, no se desea interrumpir el periodo en que el integrado tiene su salida en nivel alto, por lo que el pulsador conectado al **RESET** puede no ser necesario.

Dado que para obtener largos periodos en estado alto (superiores a los 10 minutos) se deben utilizar condensadores electrolíticos, y estos presentan fugas que afectan su confiabilidad, es que tenemos que recordar en el momento de hacer nuestros diseños que pueden ser posibles errores de hasta un 20% en los tiempos determinados por **R1** y **C1.**



Como puede verse, el esquema de nuestro oscilador monoestable es sumamente simple.

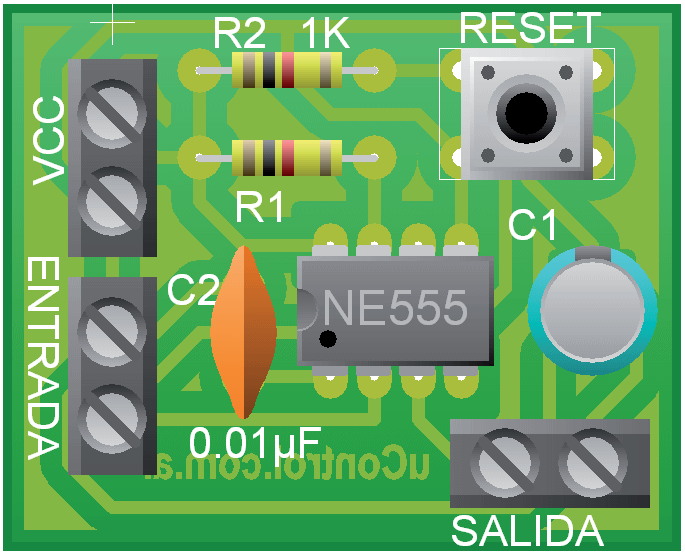


Este circuito impreso servirá para montar nuestro temporizador.

Es importante aclarar que una vez disparado el monoestable, hasta que no transcurra el tiempo T (o se resetee el temporizador) cualquier actividad en el TRIGGER es ignorada, por lo que un disparo efectuado durante el estado alto de la salida no será tenido en cuenta.

El circuito puede alimentarse con tensiones comprendidas entre 3 y 15V, y el valor de T es independiente del valor de VCC.

Las aplicaciones para este circuito son muchas. Por ejemplo, buscando los valores adecuados de R1 y C1 podría utilizarse para, mediante un relé, activar las luces de un pasillo, o la luz de cortesía de un automóvil.



Una vez montados los componentes, nuestro proyecto se verá más o menos así.

CAPACheck®

Nueva línea de comprobadores de capacitores en circuito y bobinados tipo flyback

Estos instrumentos resultan imprescindibles para detectar capacitores en mal estado, sin necesidad de desconectarlos de sus circuitos de trabajo, ni necesidad de descargarlos y aun con voltaje de corriente continua presente (en el modo AC), hasta 630 Volts DC.
La familia de instrumentos CAPACheck ha sido diseñada para ser utilizada por técnicos reparadores de equipos electrónicos y eléctricos de consumo y profesionales, en los cuales se hallan decenas y hasta cientos de capacitores electrolíticos o de otros tipos, componentes que por su tecnología de fabricación tienen una vida útil limitada comparado con otros componentes electrónicos, y que más tarde o más temprano provocarán fallas diversas en los distintos circuitos en donde estén aplicados.
Adicionalmente permiten hacer comprobaciones rápidas en bobinados de media / alta frecuencia, como son los del tipo flyback encontrados en el interior de televisores y monitores con tecnología T.R.C. para ordenadores, con lo que podrá fácilmente determinar si el bobinado en cuestión está sano o en cortocircuito.

La serie CAPACheck se presenta en dos modelos: **PLUS 911 XL**, el modelo de lujo de la mejor calidad, con funciones extra y garantía extendida, **LITE 850 XL**, el modelo más popular, económico y preferido por el técnico a domicilio y el estudiante.

Accesorios para la línea CAPACheck mod. 735 y 850

Módulo Indicador de Bajo Voltaje

Monitorea el voltaje de la batería de 9V y destella un led dentro del instrumento, visible externamente, previniendo mal funcionamiento por bajo voltaje. De fácil instalación, no necesita mecanizado.

Clonador autónomo de memorias EEPROM

//Reinier Torres Labrada//
reiniertl@gmail.com

el bus SPI

El BUS SPI (Serial Peripheral Interface) constituye un estándar desarrollado por Motorola con el objetivo de comunicar dispositivos electrónicos ubicados en la misma placa de circuito impreso. SPI establece un enlace serie sincrónico para la comunicación bidireccional, en modo full duplex, y configuración Maestro/Esclavo. Solo el Maestro puede iniciar la comunicación y se permite la existencia de varios esclavos, con dos configuraciones circuitales posibles. Este BUS también es conocido como BUS serie de 4 hilos.

.La idea fundamental tras SPI

SPI utiliza un registro de desplazamiento circular distribuido para el intercambio de datos entre el maestro y el esclavo. Con dos líneas de datos y una de reloj, es posible transmitir simultáneamente, con cada pulso de reloj, un bit del maestro al esclavo y otro bit del esclavo al maestro. Esta configuración podemos observarla en la Figura 1, y hace posible la transmisión de datos en ambos sentidos a altas velocidades.

El análisis de la Figura 1 sugiere que la longitud del registro de desplazamiento de cada uno de los circuitos involucrados en la comunicación puede ser cualquiera, siempre que ambos dispositivos tengan la misma longitud. De hecho la especificación original no establece una longitud determinada, pero por cuestiones prácticas, la más utilizada es 8 bits.

.Señales del BUS SPI

SPI establece el uso de 4 señales, cada una de ellas con su identificador y función específica:

- **SCK** (Serial Clock): Es la señal de reloj, se utiliza para sincronizar el intercambio de datos entre los dispositivos que se están comunicando. El maestro es el único que puede generar esta señal.
- **MOSI** (Master Output Slave Input): Flujo de bits que el maestro transmite al esclavo
- **MISO** (Master Input Slave Output): Flujo de bits que el maestro recibe del esclavo
- **SS** (Slave Select): Entrada de selección del esclavo, en los maestros esta señal se utiliza para el control y manejo de errores en sistemas con más de un maestro.

Además de la especificación de nombres anterio-

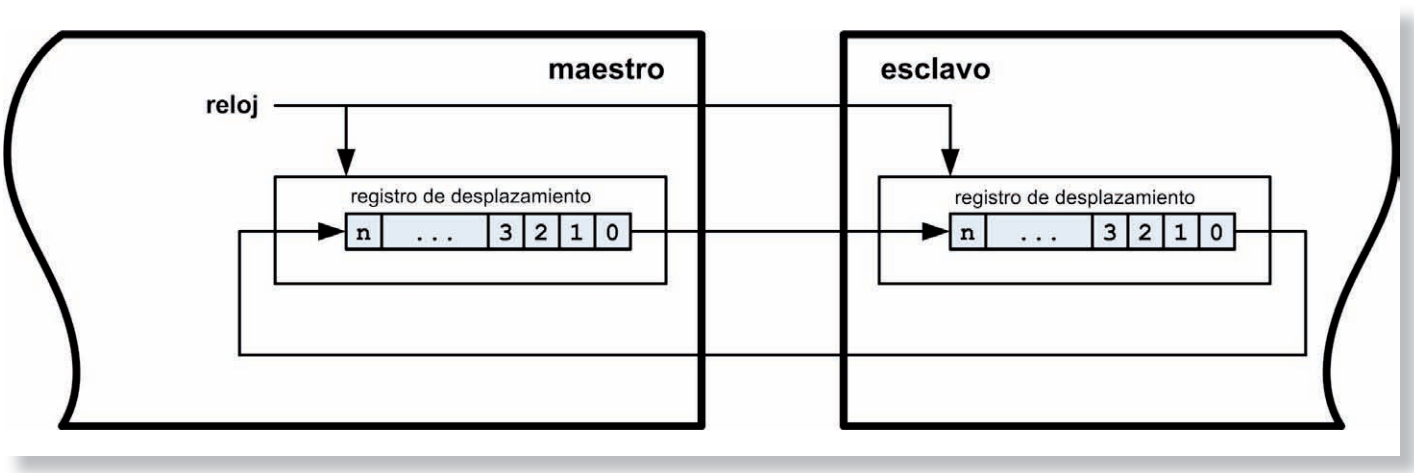


Figura 1: SPI utiliza un buffer circular distribuido para intercambiar datos entre el dispositivo maestro y el esclavo.

res existen otras ampliamente utilizadas:
- **SCK, SCL, SCLK, CLK** (Serial Clock)
- **SDI** (serial Data In), **DI** (Data In), **SI** (Serial In)
- **SDO** (Serial data Out), **DO** (Data out), **SO** (Serial Out)
- **nCS** (1..n Chip Select), **CS** (Chip Select), **nSS** (1..n Slave Select), STE (Slave Transmit Enable)

La recomendación a seguir con la denominación de las señales del BUS SPI es que se utilice la descrita en el estándar. Sin embargo cuando se necesario utilizar otra de las denominaciones mencionadas, es preciso tener en cuenta que la línea SDO del maestro debe ir conectada a SDI del esclavo y que la línea SDI del maestro debe ir conectada a SDO del esclavo, siguiendo esta recomendación para el resto de las denominaciones aceptadas. Las líneas SCK y CS cumplen la misma función que en la denominación de la especificación. La ventaja de utilizar la primera denominación descrita, radica en que las líneas de igual nombre se conectan entre sí, sin necesidad de intercambiar ninguna línea.

.SPI en detalle

Ahora que conocemos el principio básico y las señales del BUS SPI estamos en condiciones de analizar con mayor profundidad su funcionamiento. Para ello utilizaremos la Figura 2, en la cual aparecen todos los componentes fundamentales que debe tener cualquier módulo SPI.

Para nuestro análisis hemos optado por utilizar registros de desplazamiento de 8 bits, este dato será muy importante cuando veamos las cartas de tiempo de las señales en el BUS. En la figura 2 se observan nuevos elementos que completan un periférico SPI.

El primer elemento a tener en cuenta es que un periférico SPI será parte de algún circuito mayor que necesite comunicarse con otro que realiza una función diferente. Por lo tanto, hay muchos tipos de circuitos que utilizan periféricos SPI, entre ellos podemos citar: microcontroladores, memorias, puertos de comunicación para otros estándares, pantallas LCD, entre otros. El bloque con fondo amarillo claro, en la Figura 2, se corresponde con esta parte del circuito integrado.

La "unidad de control SPI", no es un circuito definido en ninguna parte del estándar, de hecho cada implementación de este bloque depende del dispositivo, y normalmente no se describe así; en este artículo lo hacemos de esta forma con fines didácticos. La "unidad de control SPI" usualmente se compone de un conjunto de registros de configuración, FIFOs y lógica de control para la generación de interrupciones.

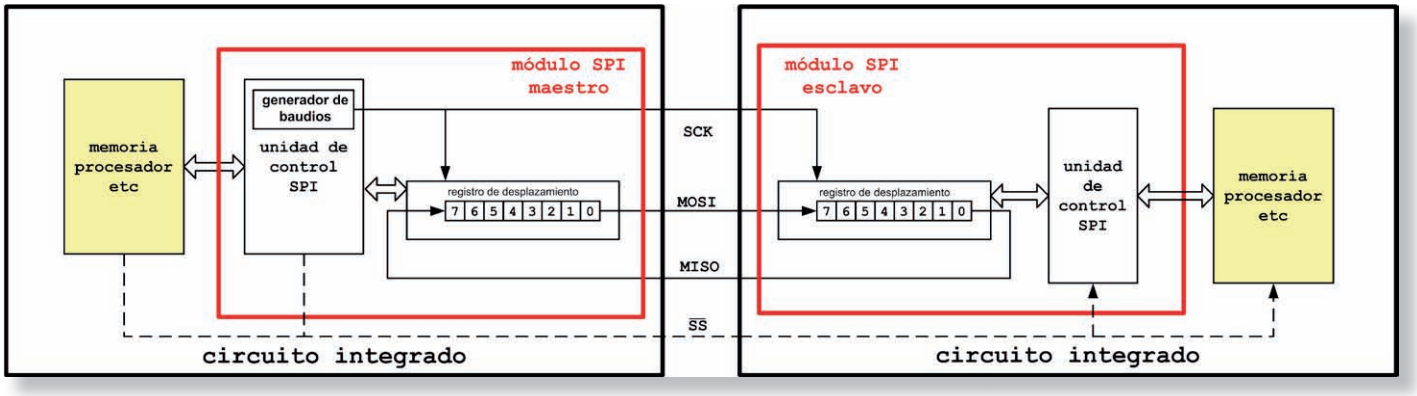
Mediante la "unidad de control SPI" el sistema que aloja al periférico tiene la posibilidad de configurarlo para el trabajo, colocar los datos a transmitir en el registro de desplazamiento y leer y guardar en un lugar seguro los datos recibidos, generar interrupciones, y en el caso del maestro, iniciar y finalizar la transferencia de datos. Para los microcontroladores con un módulo SPI dedicado, esto es especialmente útil, ya que el procesador puede dedicarse a realizar otro tipo de operaciones mientras se transmiten y reciben los datos.

SPI establece el uso de 4 señales, cada una de ellas con su identificador y función específica

En dispositivos como las memorias, la "unidad de control SPI", consiste en la implementación de una máquina de estados para algún protocolo de más alto nivel que asegura la lectura y escritura de los datos almacenados en memoria.

El único elemento que encontrará en los datasheets como parte indispensable de una implementación SPI, es el generador de baudios del maestro. El generador de baudios, es en esencia, un divisor de frecuencias programable, cuya salida es la señal de reloj del BUS.

El funcionamiento de los registros de desplazamiento es como ya se indicó previamente y la señal de selección del esclavo (SS) puede ser generada por la propia "unidad de control SPI" o por otro componente cualquiera del maestro; en los microcontroladores suelen utilizarse puertos de E/S de propósito general para este fin. Se ha dibujado esta señal con línea discontinua porque puede que el esclavo no la tenga conectada al maestro. Por ejemplo, en un sistema que tiene un maestro y un esclavo, puede que lo más conveniente sea mantener al esclavo seleccionado en todo momento, así que la línea SS del



El módulo SPI en el maestro y en el esclavo, y los bloques funcionales que permiten interactuar con el dispositivo que lo contiene.

esclavo iría directamente a tierra y solamente se utilizarán para comunicarse las líneas MOSI y MISO.

Hay situaciones en que el maestro solamente realizará un tipo de operación con el esclavo. En este caso no se requiere conectar las líneas MOSI y MISO, sino solamente una de ellas, en función del tipo de operación a realizar; existen además dispositivos con modos de comunicación bidireccional que utilizan una sola línea de datos. Pero por ser configuraciones no habituales, quedan fuera del alcance de este trabajo.

.Comunicación entre dispositivos

La transferencia de datos solamente puede ser realizada por el maestro, único dispositivo que tiene el control de la señal de reloj y que usualmente tendrá el control de la señal de selección del esclavo.

Para iniciar la comunicación con un esclavo cualquiera, el maestro debe en primer lugar, establecer la frecuencia que tendrá la señal de reloj. Para ello debe configurar adecuadamente su generador de baudios, de modo que éste genere una señal de reloj con una frecuencia igual o inferior a la frecuencia máxima que permite el esclavo.

Posteriormente el maestro pondrá a nivel lógico bajo la señal SS del esclavo con el cuál desea establecer la comunicación. Si fuese necesario esperar un tiempo determinado, por ejemplo, para que un conversor AD termine el proceso de conversión o para que el esclavo se encuentre listo para la transferencia. El maestro debe implementar algún mecanismo de demora para estos casos, antes de comenzar a transmitir o recibir datos.

El siguiente paso es transmitir y recibir un bit de datos con cada pulso de reloj, los bits son empujados por la línea MOSI, uno a uno en el registro de desplazamiento del esclavo y el mismo proceso se produce en sentido inverso por la línea MISO.

La cantidad de pulsos de reloj de cada transmisión no está limitada y depende de las características de la comunicación entre los distintos dispositivos, sin embargo las longitudes de palabra más utilizadas son 8, 12, 16 y 32 bits, por trama de datos.

Una vez terminada la transmisión el maestro deja de generar la señal de reloj y habitualmente llevará la señal SS a estado alto para desactivar la comunicación con el esclavo previamente seleccionado.

Para garantizar la comunicación en un sistema que utilice varios esclavos, se precisa el uso de dispositivos cuyas entradas y salidas sean de tercer estado, de modo que para los efectos de la comunicación, el dispositivo aparezca “desconectado” cuando su señal SS esté en estado alto.

.Los detalles del reloj y la fase

Asociados al la señal de reloj existen dos aspectos importantes que se tienen en cuenta en las implementaciones de un BUS SPI. Estos dos aspectos, la polaridad y la fase, determinan hasta cuatro modos de trabajo del BUS SPI, que cualquier diseñador debe tener en cuenta al comunicar dispositivos con este tipo de BUS.

Tomando como ejemplo un dispositivo con la posibilidad de ajustar los parámetros que permiten configurar la polaridad y la fase del reloj, este dispositivo requerirá la existencia de dos parámetros llamados CPOL (Clock Polarity) y CPHA (Clock Phase).

Polaridad: La polaridad del reloj está determinada por el estado de la salida SCK cuando el BUS está inactivo; la polaridad positiva implica que SCK estará en estado lógico alto (CPOL=1) y la negativa SCK en estado lógico bajo (CPOL=0).

Fase: Este parámetro determina con cual flanco de la señal de reloj será tomado el dato de las líneas MOSI y MISO. Para CPOL=0, el dato es leído en el primer flanco de SCK y para CPOL=1 en el segundo flanco de SCK. La lectura del dato no depende de la polaridad del reloj, sino el cambio de estado en SCK al inicio de la transferencia.

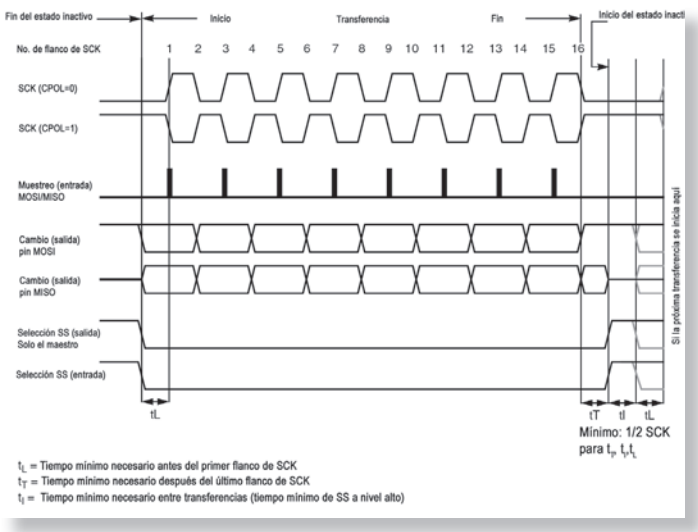


Figura 3: Carta de tiempo para el BUS SPI para CPHA=0

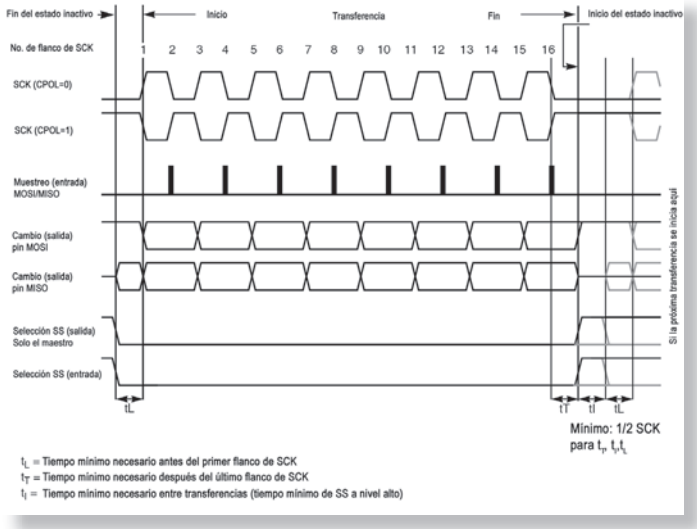


Figura 4: Carta de tiempo para el BUS SPI para CPHA=1

Para observar mejor la relación entre polaridad y fase, es preciso hacer un análisis de las Figuras 3 y 4, las cuales incluyen las cartas de tiempo, una para CPOL=0 y otra para CPOL=1. Estas figuras ofrecen detalles acerca de los estados que deben adoptar el resto de las señales presentes en el BUS y los tiempos que deben transcurrir entre un estado y otro del BUS en su conjunto.

Al analizar esta figura, podría parecer innecesario tener la posibilidad de configurar estos dos parámetros, sobre todo porque un periférico SPI configurable puede ser costoso desde el punto de vista de los recursos que demande para su implementación. Sin embargo una observación de este tipo no es adecuada, debido a que por la gran cantidad de dispositivos que pueden utilizar un sistema de comunicaciones como este, es preciso tener a mano cierta libertad en el diseño de módulos que aprovechen mejor el BUS con el objetivo de obtener un mejor desempeño del sistema.

Por ejemplo, existen conversores análogo digital (ADC) que requieren un tiempo desde el momento en que reciben la orden de convertir a digital la tensión analógica a su entrada, hasta que el dato está disponible para que otro dispositivo pueda leerlo. En algunos ADC, este tiempo puede ser muy pequeño, por lo que si utiliza SPI, bastaría con usar la señal SS para dar la orden de conversión y fase uno, de modo que el conversor tenga suficiente tiempo para terminar su tarea y tener los datos disponibles y listos para la lectura. Ejemplos como el anterior podrían citarse muchos, cada uno de ellos justificando el uso de alguno de los cuatro modos de configuración de polaridad y fase del reloj, para un tipo de aplicación particular.

.Configuraciones circuitales

Existen dos configuraciones básicas para la interconexión de dispositivos utilizando el BUS SPI, la primera de ellas es conectando cada uno de los periféricos al BUS, de modo que comparten las líneas MOSI y MISO, mientras que cada uno de ellos tiene una línea independiente para SS. Esta es la configuración que podemos observar en la Figura 5.

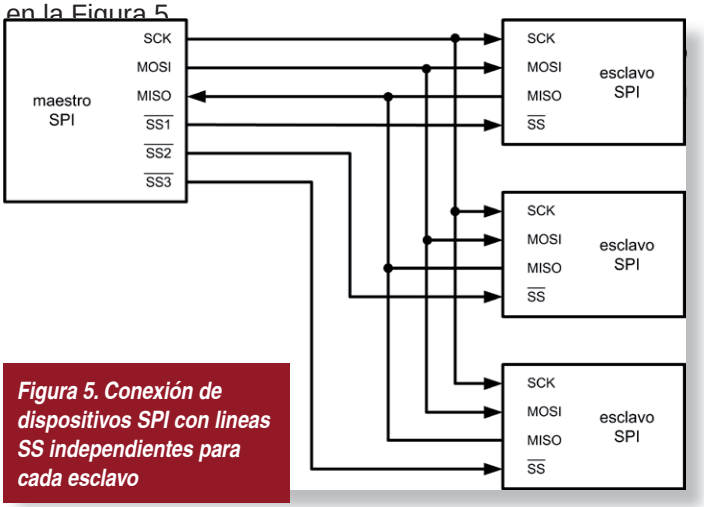


Figura 5. Conexión de dispositivos SPI con lineas SS independientes para cada esclavo

ser gobernadas directamente por el periférico SPI o tendrán que ser controladas explícitamente por el programador del sistema o por algún mecanismo hardware especialmente diseñado.

La otra configuración es en Daisy Chain, en la cual la línea MISO del primer periférico va conectada a la línea MOSI del segundo y así sucesivamente. La línea MISO del último periférico debe ir conectada a la línea del mismo nombre del maestro, todos los periféricos tienen una sola línea SS. Esta configuración es similar a tener un registro de desplazamiento muy largo, compuesto por varios periféricos SPI encadenados uno a continuación del otro. Esta configuración es la que podemos observar en la Figura 6.

Otras características que pueden encontrarse en periféricos SPI

SPI incluye varias características adicionales que permiten obtener modos de trabajo y otras prestaciones que las descritas hasta aquí. Sin embargo, su explicación queda fuera del objetivo de este trabajo además que recomendamos un estudio profundo por parte del que vaya a utilizar un dispositivo con estas características “avanzadas”.

Modo bidireccional: Permite conmutar una única línea de datos para que se comporte como MOSI o MISO, según el tipo de transferencia a realizar. Implica que el hardware se complicará aún más, pero con ello se utiliza una línea menos de datos.

Uso de SS en el maestro para entornos multimaestro: Cuando en un sistema se utiliza más de un maestro, suele utilizarse la línea SS para detectar colisiones en el acceso al BUS. El mecanismo de detección de errores que se implementa consiste en interrumpir la transferencia en proceso y notificar la colisión.

Interrupciones: Los maestros suelen implementar mecanismos de interrupción, para notificar el fin de una transferencia SPI, al procesador del sistema donde está implementado el dispositivo.

FIFOS: Algunos sistemas tienen colas de envío y recepción, de modo que se puede alimentar a la cola con una trama de datos y realizar una transferencia completa de ésta antes de notificar al procesador que es momento de vaciar el buffer del periférico.

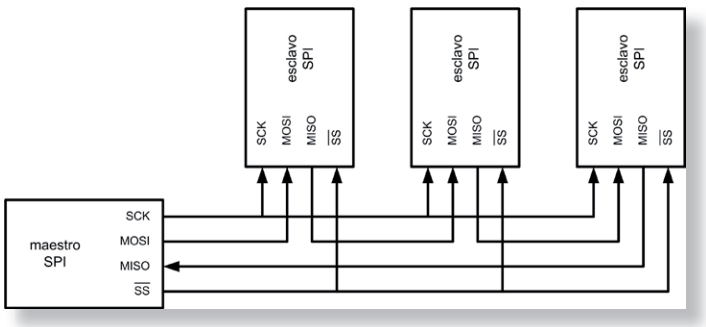


Figura 6. Conexión de dispositivos SPI encadenados (Daisy Chain)

//Mario Sacco//
service.servisystem@gmail.com

el u-scilador (ó el Oscilador Universal)

Un Oscilador de frecuencia variable, capaz de trabajar con frecuencias comprendidas entre uno y varias decenas de MHz. siempre será una herramienta muy útil en la mesa del diseñador de circuitos digitales.

¿Cuántas veces nos ha ocurrido que estamos diseñando un circuito que requiere de un oscilador para funcionar y no encontramos el circuito adecuado? Cuando encontramos uno que cubre la frecuencia deseada, chocamos con que la excursión pico a pico de su señal de salida es mayor a nuestra necesidad, ó viceversa: cuando la salida es adecuada, la frecuencia es insuficiente.

Por otro lado, quienes trabajamos con microcontroladores, necesitamos contar siempre con un stock de cristales de distintas frecuencias para poder dar vida a nuestros diseños. Más aún, cuando los puristas del ASM requieren de un desborde del TMR0 cada un tiempo exacto, necesitando por ejemplo un cristal de 12 MHz. para lograrlo, seguramente descubrirán que en todas las cajas y cajones del taller solo hay de 10 MHz. o de 14 MHz. ¡Qué desazón!

Es por eso que hemos desarrollado el u-scilador, cuyo funcionamiento explicaremos en detalle. Este oscilador podrá cubrir fácilmente cualquier frecuencia requerida para trabajar en múltiples e innumerables aplicaciones que podamos imaginar. Y todo ello lo haremos con una inversión menor a 2 Euros.

El corazón de nuestro montaje fue extraído de la página web de nuestro amigo y colaborador (de pié, por favor) Diego Marquez García-Cuervo.
Allí nos encontramos con un Multivibrador Astable sin R ni C (Resistor y Condensador, respectivamente), construido mediante un sencillo y modesto circuito integrado de la familia TTL: el 7404N.

Un Oscilador de frecuencia variable, capaz de trabajar con frecuencias comprendidas entre uno y varias decenas de MHz

El funcionamiento de éste multivibrador es muy elemental, como puede verse en el esquema. Solo debemos tener en cuenta una regla fundamental en su construcción: **la cantidad de puertas empleadas en el mismo, debe ser impar.**

Veamos cómo es que funciona este circuito. Partiendo de un estado inicial **BAJO** (ó LOW ó 0) en la entrada de la puerta A, su salida se transformará en un estado **ALTO** (ó **HIGH** o 1). Este estado digital se aplica a la segunda compuerta, la B, que cambiara su salida a un estado **BAJO**. Este nivel de tensión se aplica a la tercera puerta, que lo invierte y presenta a su salida un estado **ALTO**. La cuarta compuerta lo vuelve a invertir y se aplica su salida a la entrada de la quinta, que presentará en su salida un estado **ALTO**. Este valor se reintroducirá en la entrada de la primer puerta, invirtiendo todos los estados en las salidas **a medida que dicho cambio, vaya atravesando las puertas que forman la cadena. Cada compuerta provoca el cambio de la siguiente, como fichas de dominó que se empujan unas a otras.**

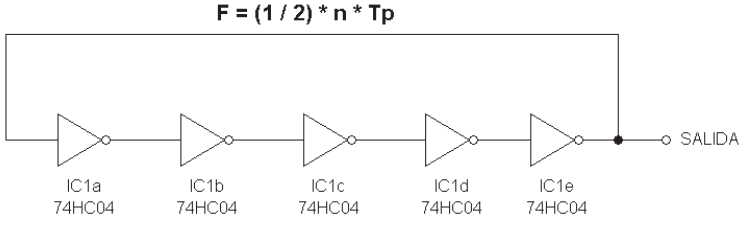
Hemos remarcado ésta última frase ya que la misma es la que nos explica el porqué de la oscilación del circuito. Es el retardo de propagación que ofrece la puerta al paso de la señal lo que determina, en parte, la frecuencia de las oscilación.

La frecuencia de oscilación se puede calcular mediante la fórmula que se observa en el gráfico:

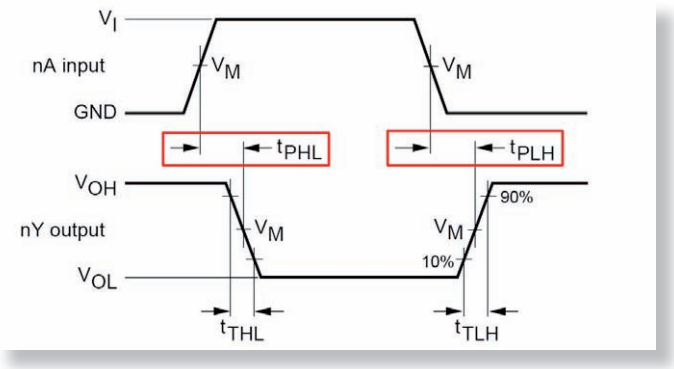
F = 1 / 2 * n * Tp

Donde **F** es la frecuencia resultante de oscilación, **n** es la cantidad de puertas y **Tp** el tiempo de propagación de la señal dentro de la puerta. O dicho de otra forma, el tiempo que demora en cristalizarse la acción a la salida de la puerta, respondiendo a la acción entrante.

En las hojas de datos de los Circuitos integrados, lo vemos de la siguiente forma:



Parece imposible que con un solo componente podamos realizar un oscilador, sin embargo, es una realidad.



tPHL nos refleja el retardo en pasar de un estado ALTO a uno BAJO y tPLH el retardo de la acción inversa, pasar de un estado BAJO a uno ALTO

Entonces, según la fórmula, obtendremos que para retardos normales de 10 nanosegundos por cada compuerta, la acumulación de las cinco nos dará:

F = 1 / 2 * 5 * 10-8

Lo que resultará en una frecuencia de 10 Mhz.

La diversidad de tecnologías de fabricación de circuitos integrados nos ofrece un vasto espectro de elección, entre los que encontramos la familia **CMOS 74HC** ó **74HCT**. Con ellos lograremos la particularidad que buscamos cumpla nuestro **u-scilador** (cualquier relación con **uControl** es mera coincidencia)

Estas pequeñas y económicas maravillas nos permiten trabajar con tensiones que van desde los 2 a los 6 Voltios de alimentación, con la particularidad de que la velocidad de propagación dentro de las puertas no será la misma para distintas tensiones. Es decir, a mayor tensión, mayor velocidad de propagación.

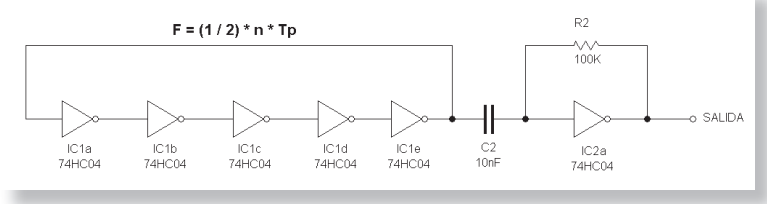
SYMBOL	PARAMETER	TEST CONDITIONS		MIN.	TYP.	MAX.	UNIT
		WAVEFORMS	V_{CC} (V)				
T_{amb} = 25 °C							
t_{PHL}/t_{PLH}	propagation delay nA to nY	see Figs 6 and 7	2.0	–	25	85	ns
			4.5	–	9	17	ns
			6.0	–	7	14	ns
t_{THL}/t_{TLH}	output transition time	see Figs 6 and 7	2.0	–	19	75	ns
			4.5	–	7	15	ns
			6.0	–	6	13	ns

La hoja de datos del IC, nos muestra la gran diferencia de velocidad a distintas tensiones de alimentación.

Por lo tanto si al primer circuito lo alimentamos con una tensión variable entre 2 y 6 volts tendremos una variación de frecuencia muy amplia a la salida del circuito, la que dependerá de cada caso en particular del IC que utilicemos. Según las pruebas y ensayos realizados en los laboratorios de **uControl**, para circuitos 74HC, hemos logrados valores de frecuencias comprendidas entre los 500 Khz. Y algo más de los 30 Mhz. ¿Asombroso, verdad?

De ésta forma obtenemos un VCO (Voltaje Controlled Oscillator, u Oscilador Controlado por Voltaje), al que sólo necesitamos agregarle una etapa más a la salida. Esta última etapa deberá tener una tensión de alimentación constante, adecuada para la amplitud de la señal que quiéramos obtener.

Debido a que la salida de la cadena osciladora no será de amplitud constante, se hace trabajar a U2 en modo de amplificación gracias a R1, para obtener amplificación a señales bajas



U1 será alimentado con una tensión variable de entre 2 y 6 Volts, mientras que U2 llevará alimentación constante.

Tenemos entre manos un oscilador que se puede construir con las monedas que tengamos en nuestro bolsillo (algo menos de 2 Euros), que cubre la gama que va desde 500 Khz. a 30 Mhz. y que en los próximos números de uControl mejoraremos con la adición de un divisor de frecuencia programable y un detector de fase. Se transformará así en un completo sistema PLL (Phase-Locked Loop o Lazo de seguimiento de fase) que nos permitirá obtener la frecuencia que se nos ocurra con una precisión casi absoluta.

No querrás perderte los próximos bytes de u-Control.

GOSUB 11

Electrónica
Controladoras CNC
eca55.com.ar
www.esteca55
PIC'S
CNC
PIC'S
Máquinas
Electrónica
Máquinas CNC
www.esteca55.com.ar

//Ariel Palazzesi//
arielpalazzesi@gmail.com

LUDOTECA

El hombre ha incorporado los juegos como parte de cultura desde el comienzo mismo de la civilizacion. Cuando jugamos, en realidad, estamos aprendiendo. Estamos entrenando nuestro cerebro. Como los juegos son tan importantes en la vida de las personas, es que en uControl hemos decidido inaugurar una nueva seccion dedicada completamente a ellos. Número a número iremos viendo como construir una serie de circuitos practicos relacionados con el tema. Bienvenidos a la ludoteca

dado electrónico (versión 1)

El primer proyecto de la "Ludoteca uControl" es un económico dado digital, basado en componentes discretos. Se trata del complemento idea para muchos juegos de mesa y se puede armar en un solo un par de horas.

Se trata de un proyecto ideal para los que están descubriendo el hobby de la electrónica, ya que su principio de funcionamiento es muy sencillo, y la dificultad de construcción baja.

El circuito se basa en un par de circuitos integrados de bajo costo (menos de un euro cada uno). El primero de ellos es el CD4017B, un contador/divisor por 10, que también puede ser configurado para contar hasta cualquier número menor que ese. El segundo es otro circuito integrado de tecnología CMOS, el CD4093B, que contiene cuatro compuertas NAND (Schmitt Trigger). Con una de estas compuertas se configura un oscilador que generara los pulsos que se encargara de contar el CD4017B. El objetivo del circuito es generar un número al azar entre 1 y 6, y mostrar el resultado en un display conformado por 7 diodos LED, uno por cada punto de los que encontramos en las caras de los dados convencionales.

La forma más simple de obtener un número aleatorio es generar un tren de pulsos de alta frecuencia, y enviarlos a un circuito contador/divisor por seis. Este tren de pulsos tiene una duración que depende del tiempo que el usuario presione el pulsador de disparo. Debido a que se generan varios miles de pulsos por segundo, es humanamente imposible anticipar el resultado de la cuenta o buscar un valor determinado modificando el tiempo que se presiona SW1.

Los pulsos se generan con el bloque funcional construido alrededor de la compuerta IC2a, el resistor R1 y el condensador cerámico C1, cuando el usuario presiona SW1. Estos pulsos se aplican a la entrada de CLOCK del CD4017B, que por cada pulso recibido incrementa en uno el valor de la cuenta, poniendo en alto la salida correspondiente. Cuando la cuenta llega a 6, el CD1017B se resetea a si mismo (la salida 7 esta conectada al pin RESET) y comienza de nuevo a contar desde uno. Esto se repite miles de veces por segundo, todo el tiempo que el jugador mantenga presionado SW1.

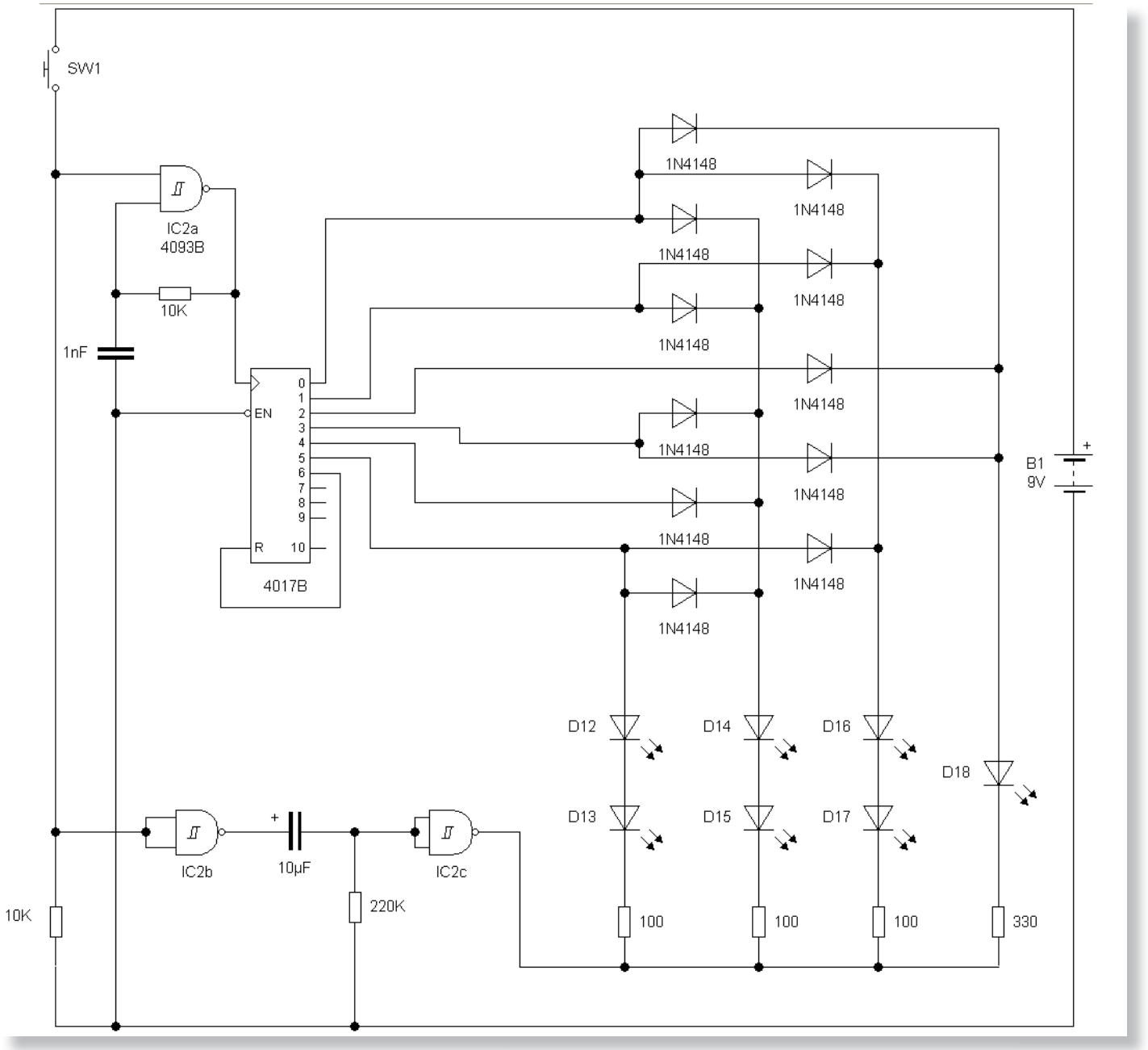
Las salidas del CD4017B son las encargadas de encender los LED que hacen las veces de display. Los demás diodos, todos 1N4148, cumplen la función de permitir que en cada caso se iluminen los LED apropiados.

Si analizamos detenidamente las caras de un dado, podemos ver que en realidad los seis valores posibles se pueden conformar con cuatro grupos de LED, tal como se ve en una de las figuras que acompañan el artículo. El LED D18 será el que ubicaremos en el centro del display (posición A), los LED D16 y D17 forman el grupo B, D14 y D15 el grupo C y D12 y D13 el grupo D. Los resistores R2 a R5 son las encargadas de limitar la corriente que circula por los LED, de forma que se iluminen todos con la misma intensidad.

El circuito se alimenta a partir de una pila de 9V, que nos proporcionara muchas horas de diversión. No obstante, y con el objetivo de preservar su vida útil, se han utilizado las otras compuertas existentes en el CD4093B para configurar un temporizador. Este se encarga de apagar el display después de unos 10 segundos. Este tiempo se determina mediante el valor del condensador C2 y el resistor R6. El lector puede experimentar con otros valores para estos componentes y modificar así el tiempo que permanece encendido el dado.

Las salidas del CD4017B son las encargadas de encender los LED que hacen las veces de display

Respecto de la construcción, comenzaremos fabricando la placa de circuito impreso de la manera que ya hemos explicado en la revista número 1, para luego montar los componentes. La placa es de doble faz, así que debemos ser cuidadosos al “planchar” cada una de ellas, de forma que los pads de uno y otro lado coincidan. Los LED deben quedar más altos que el resto de los componentes, para que se vean fácilmente. Si el lector lo prefiere, puede dotar al de una caja adecuada, dejando siete agujeros de 5mm para

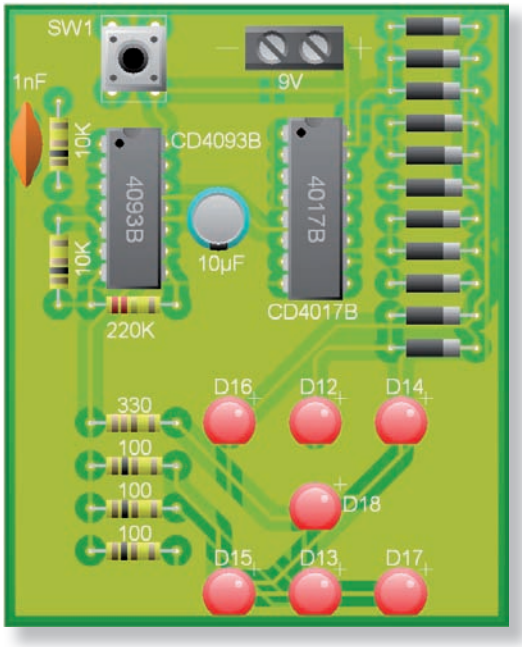


Este es el circuito de nuestro dado electrónico.

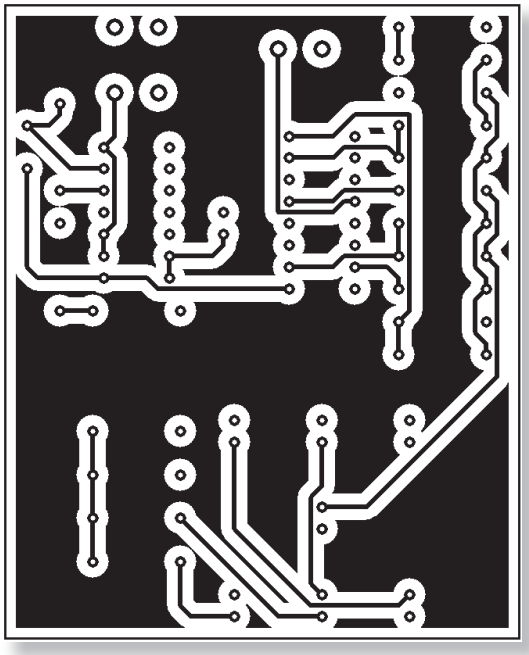
los LED. En este caso, también se debería utilizar un pulsador diferente, ya que se montaría en la caja en lugar ir soldado directamente sobre el PCB.

Lista de materiales:

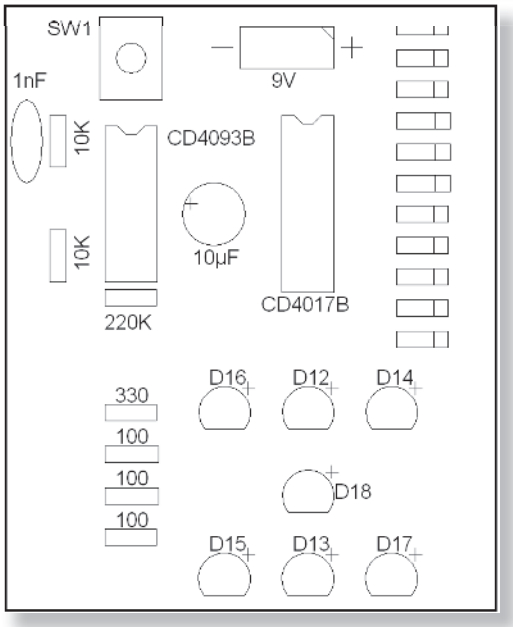
- 1 circuito integrado CD4017B
- 1 circuito integrado CD4093B
- 11 diodos 1N4148
- 7 diodos LED de cualquier color
- 2 resistores de 10K, ¼ watt
- 1 resistor de 220K, ¼ watt
- 3 resistores de 220 ohms, ¼ watt
- 1 resistor de 330 ohms, ¼ watt
- 1 condensador cerámico de 1 nF
- 1 condensador electrolítico de 10 uf/16V
- Pulsador, clip para batería de 9 volt, etc.



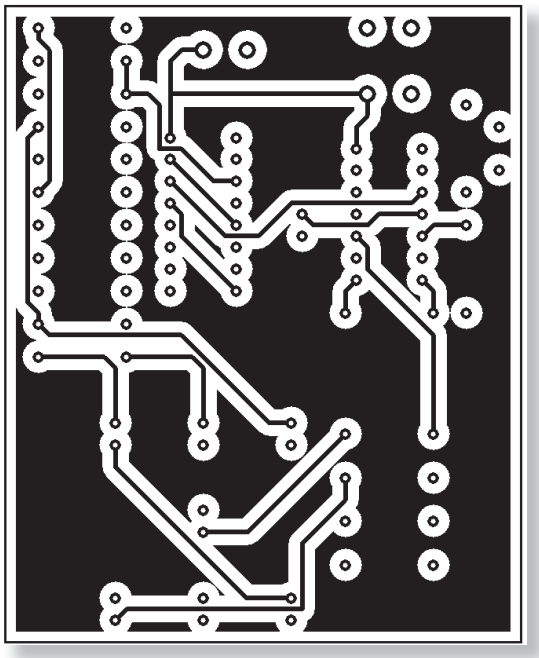
Este será el aspecto el proyecto terminado.



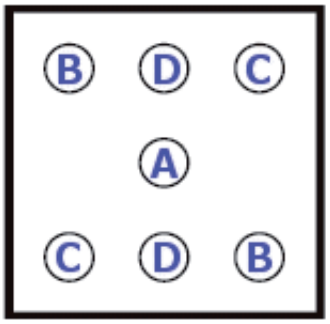
Diseño del PCB, correspondiente al lado de las soldaduras.



Ayuda para el montaje de los componentes.



Este es e diseño del PCB, lado componentes.



Salida	Grupo	LEDs
1	A+B+C	
2	B+C	
3	A	
4	A+C	
5	C	
6	B+C+D	

Con solo cuatro grupos de LEDs podemos mostrar los 6 valores posibles.

//Ariel Palazzesi//
arielpalazzesi@gmail.com

dado electrónico con PIC16F627A

Ya vimos como montar un dado utilizando componentes convencionales. Ahora le proponemos al lector el ejercicio de realizar un circuito equivalente, pero empleando un microcontrolador.

Siguiendo con los montajes destinados a formar parte de nuestra "ludoteca", presentamos un dado un tanto especial. Si bien se trata de un circuito que, como ya vimos, puede resolverse mediante una serie de componentes comunes, su simplicidad lo hace especialmente interesante para aprender a utilizar microcontroladores.

Como corazón de nuestro montaje hemos elegido a un microcontrolador de la familia **"PIC16F"** de Microchip. Concretamente, se trata del modelo **PIC16F627A,** de 18 pines. Seguramente el lector mas avanzado se preguntará el porque de esta elección, ya que, por ejemplo, un mucho mas pequeño y modesto **PIC12F675** hubiese bastado para cumplir esa función.

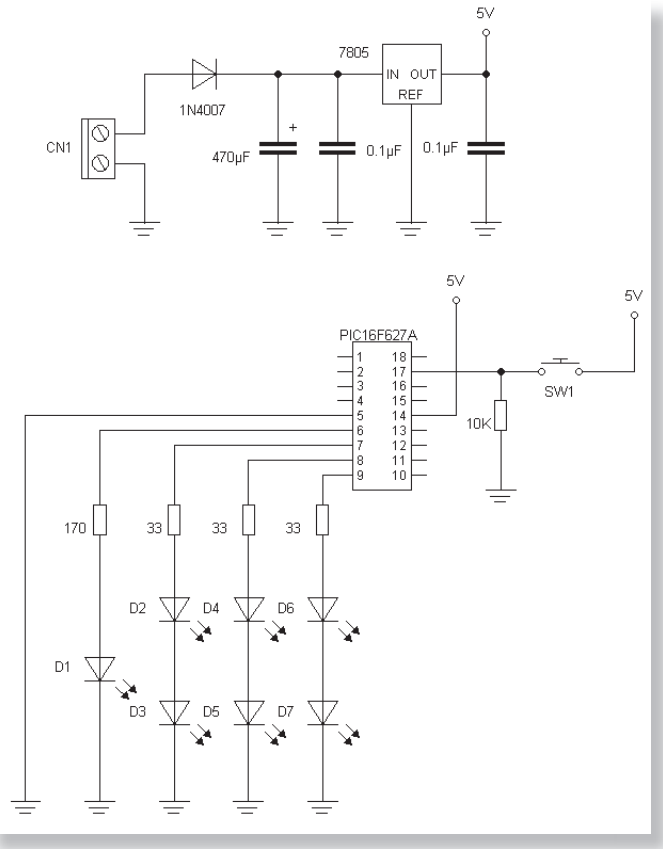
Hemos basado nuestra decisión en el hecho de que ambos microcontroladores son económicos, y cuestan prácticamente lo mismo. Y como en futuras entregas ampliaremos este montaje a un número mayor de dados, nos vendrá bien ir conociendo este PIC.

.El circuito

El circuito se destaca por su simplicidad. Podemos ver su esquema en la figura 1, donde rápidamente notamos la existencia de una pequeña fuente de alimentación construida alrededor de un regulador de voltaje de la familia **LM78xx,** concretamente un **LM7805**. Como se aconseja en la hoja de datos de este regulador, hemos incluido un par de condensadores cerámicos de 0.1 uF, uno en su entrada y otro en su salida.

Un diodo se encarga de evitar la "catástrofe" de conectar la alimentación en forma invertida, y un condensador electrolítico de **470 uF** ayuda a eliminar el ripple que pudiese existir en alimentación externa.

El resto del circuito es el dado propiamente dicho. Si has leído el articulo "PICs y LEDs" en este mismo número, seguramente el resto de este párrafo no es para ti. Cuatro salidas del PIC, concretamente RB0, RB1, RB2 y RB3, se encargan de controlar los cuatro grupos de LEDs que conforman el dado. Otras tantas resistencias limitan la corriente que circula por cada rama. La figura 2 muestra que salidas debemos activar para cada valor posible del dado.



Este es el esquema eléctrico de nuestro montaje.

Cuatro salidas del PIC se encargan de controlar los cuatro grupos de LEDs que conforman el dado.

NUMERO	SALIDAS	LEDS
1	B0 = 1 B1 = 0 B2 = 0 B3 = 0	•
2	B0 = 0 B1 = 0 B2 = 0 B3 = 1	• •
3	B0 = 1 B1 = 0 B2 = 0 B3 = 1	• • •
4	B0 = 0 B1 = 1 B2 = 0 B3 = 1	• • • •
5	B0 = 1 B1 = 1 B2 = 0 B3 = 1	• • • • •
6	B0 = 0 B1 = 1 B2 = 1 B3 = 1	• • • • • •

Grupos a encender para cada valor del dado.

Un pulsador, conectado a RA0, se encarga de poner ese pin a +V cada vez que se lo presiona. Esto nos permite saber que el usuario ha decidido lanzar el dado. Un resistor de 10k mantiene RA0 conectado a GND el resto del tiempo.

.El programa

El programa necesario para que nuestro proyecto realmente funcione es bastante simple. Se limita a esperar que el usuario presione el pulsador, y mientras que lo mantiene en esa condición cuenta continuamente de 1 a 6.

Hemos escrito dos versiones del programa: una en PIC BASIC y otra en CCS.

El tiempo que el jugador mantiene presionado el pulsador es el que determina el estado final del dado. El hecho de que el PIC sea capaz de hacer esto decenas de miles de veces por segundo nos garantiza la imposibilidad de que el usuario pueda influir conscientemente en el resultado.

Igual que en otras oportunidades, hemos incluido una versión del programa en PIC BASIC, y otra en CCS:

```
'-----CONFIGURAMOS PUERTOS----------
AllDigital  'Todos los pines del PORTA como E/S
TRISA = 0xff  'Configuro el PORTA completo como entrada.
TRISB = 0x00  'Configuro el PORTB completo como salida.

'------DECLARO VARIABLES -----------
Dim i As Byte  'i contendrá el valor del dado en cada momento.

loop:  'bucle principal
       PORTB = 0  'Apago todos los leds

       If PORTA.0 = 1 Then  'Si el pulsador esta en alto...
              Gosub lanzo_dado
       Endif
Goto loop

End  'Fin del programa principal

'-------------------------------------------------
'Esta rutina lanza el dado y muestra el resultado
'-------------------------------------------------
lanzo_dado:
       i = 0  'Valor inicial de i
       While PORTA.0 = 1  'bucle principal
              i = i + 1  'Sumo 1 a i

              If i = 7 Then  'si pasa de 6...
                     i = 1  '..empiezo de 1 otra vez.
              Endif
       Wend

       'Ahora, muestro el valor de "i" en los leds:
       PORTB = LookUp(0x01, 0x08, 0x09, 0x0a, 0x0b, 0x0e), i
       WaitMs 5000  'espero 5 segundos, y vuelvo al programa ppal.
Return
Epígrafe: Software en versión PIC BASIC.

#include <16f627a.h>           //PIC utilizado
#fuses XT,NOWDT,NOPROTECT,PUT  //Configuramos los fuses
#use delay (clock=4000000)     //Oscilador a 4Mhz
#use fast_io(b)                //Optimizamos E/S del PORTB

#BYTE PORTB = 0x06             //El PORTB está en 0x06

//Variables globales-----------------------------------------------
//La tabla dado tiene el valor del PORTB para cada numero posible.
int dado [6]={0x01,0x08,0x09,0x0A,0x0B,0x0E};

//Prototipo de funciones -------------------------------------------
void lanzo_dado(void);
```

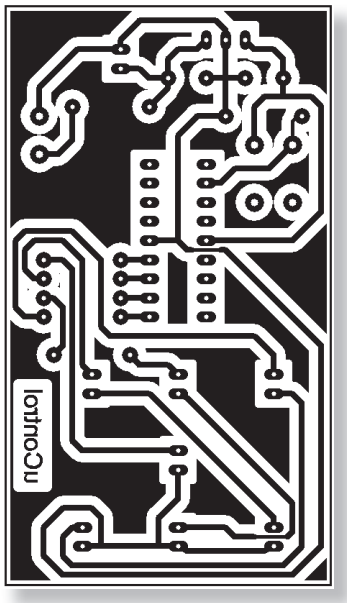
```
//------Programa principal--------------------------------------------
void main(void)
{
   set_tris_a(0xFF);           //configuro el PORTA completo como entrada.
   set_tris_b(0x00);           //configuro el PORTB completo como salida.

   disable_interrupts(GLOBAL); //todas las interrupciones desactivadas

   do{                         //bucle principal
      PORTB = 0;               //Apago todos los leds
      if(input(PIN_A0)){       //Si elpulsador esta en alto...
            lanzo_dado();      //...lanzo el dado y muestro el resultado
      }
      }while(TRUE);            //Repito el bucle.
}
//---------------------------------------------------------------------
//Esta funcion lanza el dado y muestra el resultado
//---------------------------------------------------------------------
void lanzo_dado(void) {
      char i=0;             //i contendrá el valor del dado en cada momento.

      do{                   //bucle principal
         i++;               //Sumo 1 a i

         if (i == 7) {      //Si pasa de 6...
            i = 1;          //...empiezo de 1 otra vez.
         }
      }while(input(PIN_A0)); //Repito el bucle hasta que se suelte el pulsador.

      //Ahora, muestro el valor de "i" en los LEDs:
      PORTB =  dado [i];
      delay_ms(5000);         //Espero 5 segundos, y vuelvo al programa ppal.
}
```

Este es el programa, escrito en CCS.

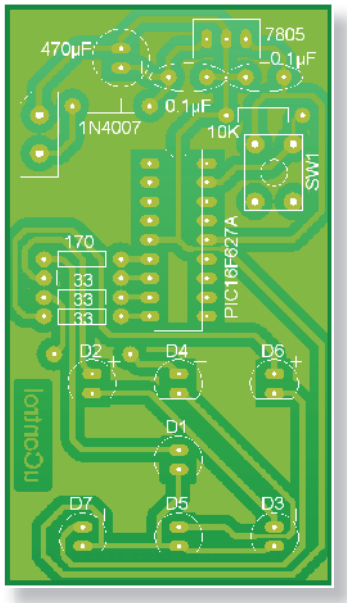
En *www.ucontrol.com.ar* encontrarás los archivos **.BAS** y **.C,** además de ambos **.HEX** listos para cargar en el **PIC16F627A.**

.El montaje

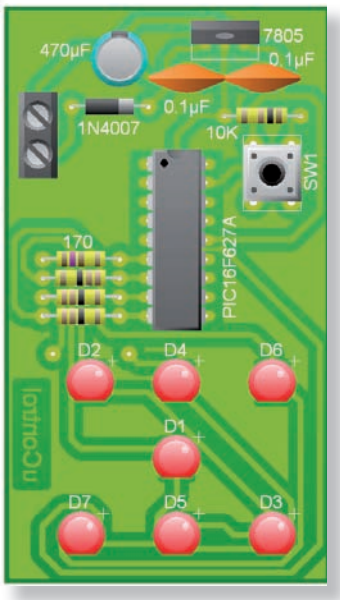
Se trata de un proyecto pequeño, así que no tendrás problemas para montarlo. En la figura 3 puedes ver el diseño del PCB que hemos realizado para alojar los componentes (y que puedes descargar a escala de www.ucontrol.com.ar).



PCB propuesto para este montaje.



Este es el aspecto del PCB, listo para comenzar a soldar los componetes.



Este será el aspecto del dado terminado.

Las figuras 4 y 5 muestran el PCB sin componentes y ya terminado. Ambas imágenes pueden ser de ayuda a la hora de soldar los componentes. Te recomendamos que prestes atención a la hora de colocar los LEDS, el diodo, el regulador de voltaje y el zócalo del PIC16F627A sobre el PCB, ya que si alguno de ellos queda en la posición incorrecta, el dado no funcionará.

//Ariel Palazzesi//
arielpalazzesi@gmail.com

Paleotrónica

el microprocesador Intel 4004

Si bien no se trata de un microcontrolador, la historia del microprocesador Intel 4004 tiene todos los ingredientes necesarios para que resulte interesante a cualquier amante de la electrónica. La historia del que sería el primer chip microprocesador comienza en el año 1969, en el Silicon Valley, ese mítico lugar, ubicado en el estado de California, que ya en esa época era el centro de la industria de los semiconductores.

La historia comienza cuando una joven empresa japonesa, dedicada a la fabricación de calculadoras, decidió contratar a Intel (que había sido fundada el año anterior) para que diseñara el conjunto de chips que daría vida a su nuevo modelo de calculadora electrónica de mesa. Busicom, quería mantener los costos de producción de su calculadora lo más bajos posibles.

Luego de analizar el problema, los ingenieros de Intel quedaron convencidos que el trabajo era todo un reto. Intel no estaba preparada para realizar circuitos integrados custom ("a medida") de la complejidad solicitada. Por otra parte, la cantidad de cada chip que necesitaba Busicom era bastante pequeña, lo que aumentaba el costo de cada lote. Un posible camino a la solución del problema vino de Marcian Edward Ted Hoff (Jr.), que se desempeñaba como jefe del departamento de investigación de aplicaciones, en Intel.

> El Intel 4004 es un artículo muy buscado por los coleccionistas. Suelen pagarse hasta U$S 400

Durante el otoño del hemisferio norte, en 1969, Hoff, ayudado por Stanley Mazor, definieron una arquitectura que consistía en un CPU de 4 bits, una memoria ROM para almacenar las instrucciones de los programas, una RAM para almacenar los datos y algunos puertos de entrada/salida para la conexión del teclado, la impresora, los interruptores y el display. Además definieron y verificaron el conjunto de instrucciones con la ayuda de ingenieros de Busicom, cuyo equipo lideraba Masatoshi Shima.

En abril de 1970 Federico Faggin se sumó al staff de Intel.Debía concentrarse en terminar el conjunto de chips de la calculadora de Busicom. Se suponía que Hoff y Mazor habían completado el diseño lógico de los chips y solamente restaba definir los últimos detalles, para iniciar el proceso de producción. Sin embargo, esto no fue lo que Faggin encontró cuando comenzó a trabajar en Intel ni lo que Shima encontró cuando llegó desde Japón.

Shima esperaba llegar a EE.UU, juntarse con los ingenieros de Intel, revisar la lógica de diseño para confirmar que Busicom podría tener al fin su calculadora, y regresar a Japón. Nada de esto fue posible. Se puso furioso cuando vio que estaba todo igual que cuando se había ido seis meses antes, así que utilizando su rudimentario inglés dejo claro que "No hay nada para revisar. Esto es sólo idea". Los plazos del contrato entre Intel y Busicom no se habían cumplido.

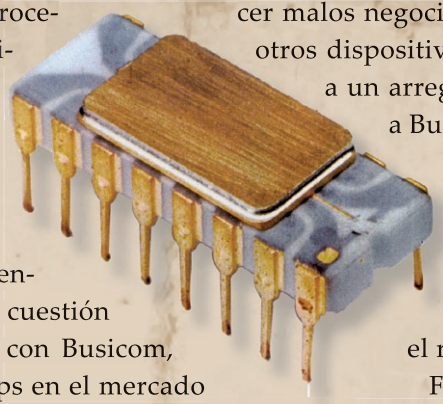
Todo esto forzó a Faggin a mantener jornadas de trabajo de 12 a 16 horas por día durante varios meses. Finalmente pudo realizar los cuatro chips mencionados, a los que denominó "la familia 4000". (Ver recuadro)

El 4001 fue el primer chip de la familia en diseñarse y construirse. El primer lote salió de la línea de fabricación en octubre de 1970 y funcionó perfectamente. En noviembre salieron el 4002 (con un pequeño error) y el 4003 (que funcionó correctamente). Finalmente, unos pocos días antes del final de 1970, nació el 4004.

Lamentablemente, en la fabricación se habían olvidado de poner una de las máscaras y toda la partida se descartó. Tres semanas después vieron la luz los nuevos 4004, con lo que Faggin pudo realizar las verificaciones. Sólo encontró unos pequeños errores. En febrero de 1971 el 4004 funcionaba correctamente. En el mismo mes recibió de Busicom las instrucciones que debían ir grabadas en la ROM.

A mediados de marzo de 1971, envió los chips a Busicom, donde verificaron que la calculadora funcionaba perfectamente. Cada calculadora necesitaba un 4004, dos 4002, cuatro 4001 y tres 4003. Tomó poco menos de un año desde que concibieron la idea hasta que el producto final funcionó correctamente.

Luego de que el primer microprocesador fuera una realidad, Faggin le sugirió a los gerentes de Intel que utilizara esa familia de chips para otras aplicaciones. Sin embargo, los directores de la empresa suponían que los chips diseñados recientemente solo servían para calculadoras, por lo que la sugerencia fue rechazada. Además, estaba la cuestión del contrato de exclusividad firmado con Busicom, que solo permitía a Intel poner los chips en el mercado con los japoneses como intermediarios.

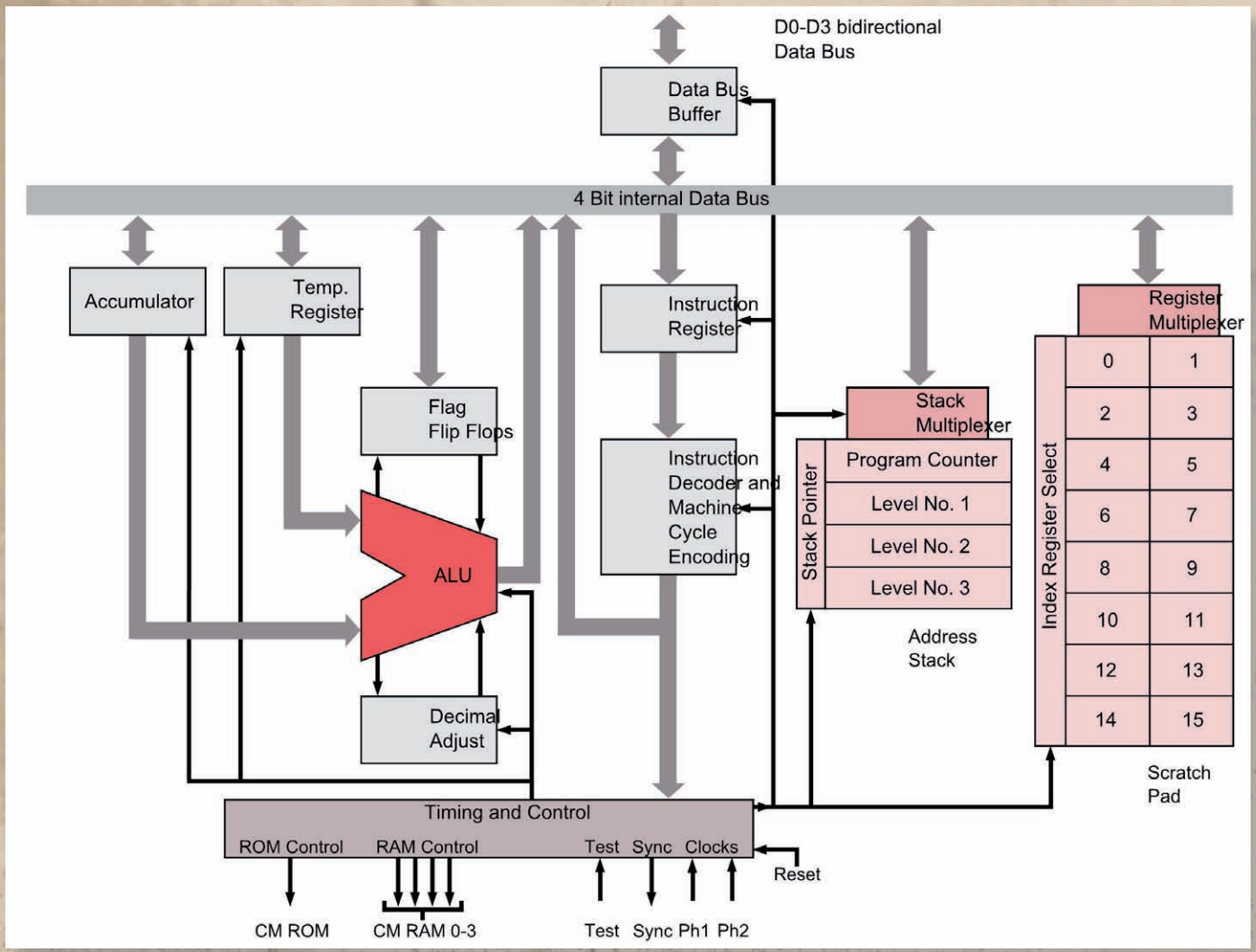


Dado que en Intel no se caracterizan por hacer malos negocios, y luego de que Faggin construyera otros dispositivos utilizando la familia 4000, llegaron a un arreglo con los japoneses: Intel le devolvió a Busicom los 60.000 dólares que había costado el proyecto, y recuperó los derechos sobre la familia 4000. Sólo podría vender los integrados para aplicaciones que no fueran calculadoras, y Busicom los obtendría más baratos que el resto de los clientes.

Finalmente, el 15 de noviembre de 1971, la familia 4000, luego conocida como MCS-4 (Micro Computer System 4-bit) fue finalmente introducida en el mercado.

Descripción del 4004

Técnicamente, el Intel 4004 es un microprocesador con un de bus de datos de 4 bits, capaz de direccionar 32.768 bits de ROM y 5.120 bits de RAM. Puede direccionar 16 líneas de entrada (de 4 bits) y 16 de salida (de 4 bits). Contiene alrededor de 2300 transistores MOS de canal P de 10 micrones. El ciclo de instrucción es de 10,8 microsegundos.



Esta era la estructura iterna del primer microprocesador

Chips de soporte

Tal como ocurre con los microprocesadores actuales, el Intel 4004 necesitaba de una serie de chips de soporte o “chipset” para funcionar. Este grupo de circuitos integrados estaba formado por 5 chips, que en conjunto se los conocía como “la familia 4000”:

-> Intel 4001: Una ROM de 256 bytes (256 instrucciones de programa de 8 bits), y un puerto incorporado de I/O de 4 bits (*)

-> Intel 4002: Una RAM de 40 bytes (80 palabras de datos de 4 bits), y un puerto de salida incorporado de 4 bits. La RAM del chip está organizada en cuatro registros de veinte palabras de 4 bits:

16 palabras de datos (utilizadas para los dígitos de mantisa en el diseño original de la calculadora)

4 palabras de estado (utilizadas para los dígitos de exponente en el diseño original de la calculadora)

-> Intel 4003: Un registro de desplazamiento de salida paralela de 10 bits para explorar teclados, pantallas, impresoras, etc.

-> Intel 4008: Un latch de 8 bits de dirección, para facilitar el acceso a chips de memoria estándar, y un chip incorporado de 4 bits de selección y puerto de I/O (*)

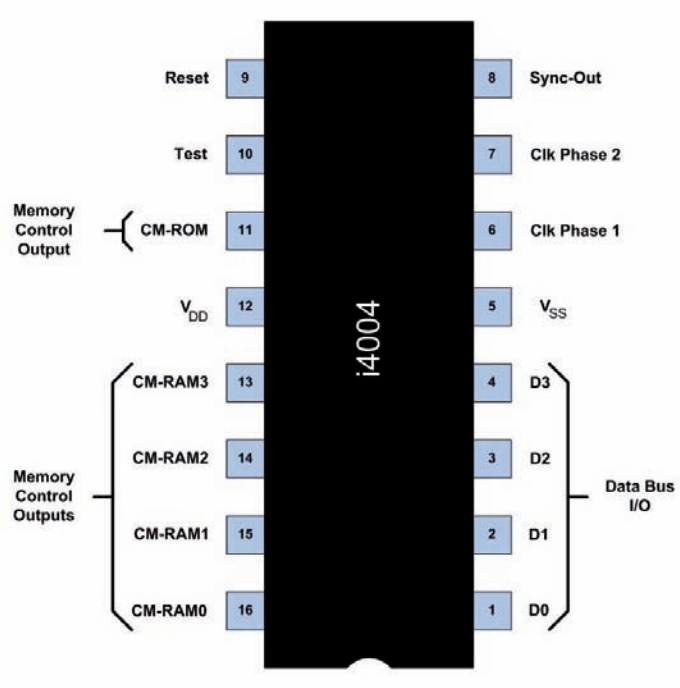
-> Intel 4009: Interfaz de acceso I/O a memoria estándar y a chips de I/O (*)

(*) Nota: Debido a limitaciones técnicas, no era posible utilizar a la vez un chip 4001 junto a un 4008/4009.

Curiosidades

-> Federico Faggin coloco sus iníciales en el centro del sustrato de la primera versión del Intel 4004. En versiones posteriores, llevadas a cabo algunos meses más tarde, el rediseño de la zona del chip que contenía la “firma” hizo que las “FF” fuesen a parar para siempre al borde del integrado.

->El Intel 4004 es un articulo muy buscado por los coleccionistas. Los de mayor valor (unos U$S 400 en eBay) son los de color oro y blanco, con los llamados “trazos grises” en la zona blanca. Le siguen los que carecen de esas marcas (U$S



Disposision de pines del Intel 4004

Bibliografia:
Darío Alpern, en http://www.alpertron.com.ar/4004.HTM
The Intel 4004, en http://www.intel4004.com/
Wikipedia, http://es.wikipedia.org/wiki/Intel_4004

MICROCONTROLADOR PIC16F84. Desarrollo de proyectos

Ra-Ma Editorial

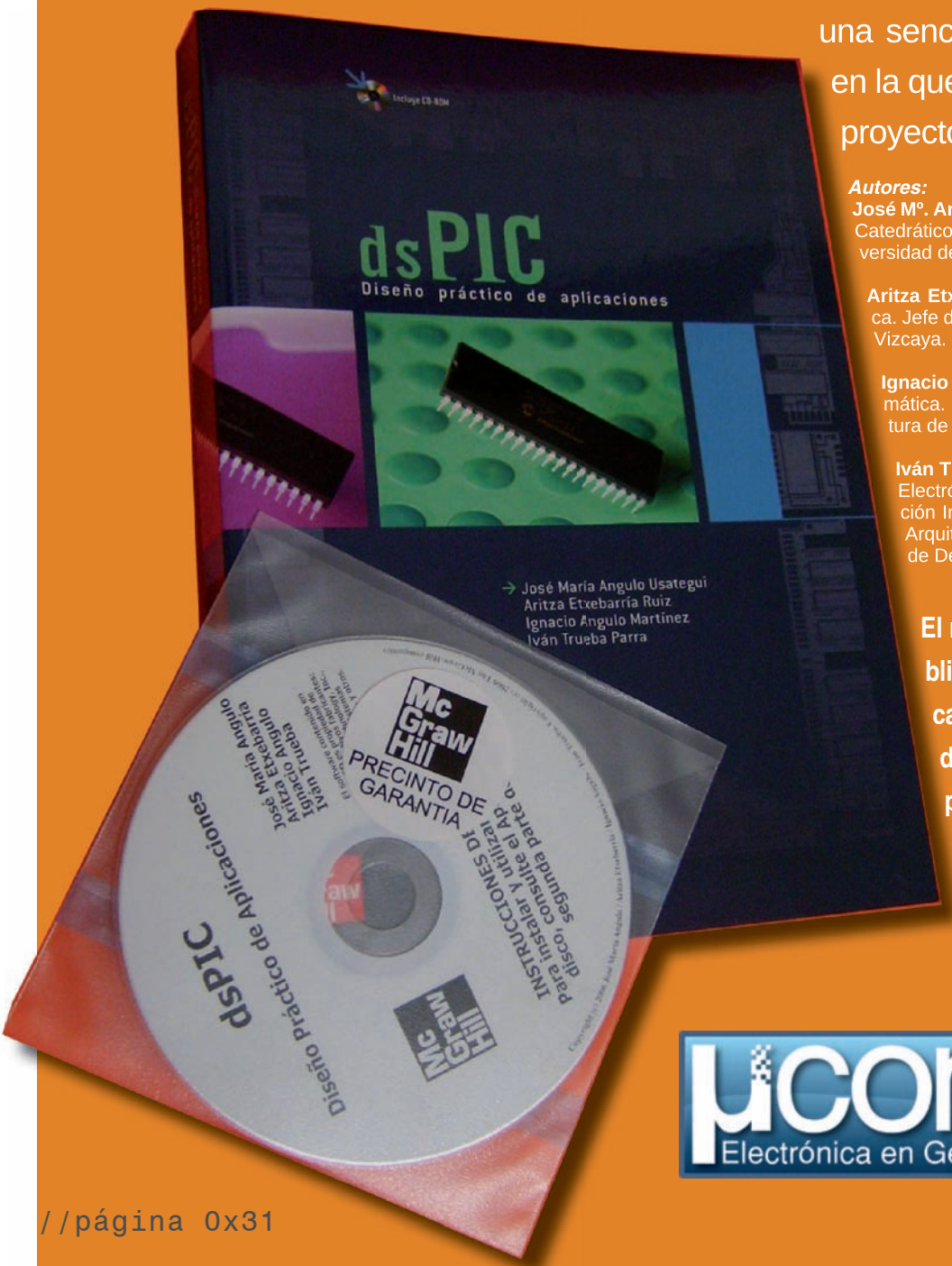
FOROSELECTRONICA.ES

la comunidad online dedicada a temas de electrónica

¡¡SORTEO!!

Los lectores de este número están de parabienes. Con solo enviar un mail con su **nombre**, **edad** y **nacionalidad** a ***sorteo@ucontrol.com.ar*** citando el numero de esta pagina, puedes ser el ganador del libro **"dsPIC, Diseño practico de aplicaciones"**, de 409 paginas. El libro, por supuesto, esta en español y se acompaña de un **CD-ROM** conteniendo el código fuente explicado en el texto.

Se trata de un libro claro, eficaz y práctico, que constituye una extraordinaria herramienta para dominar los microcontroladores PIC de nivel superior, con los que es posible encarar aplicaciones propias de los DSP. Se encuentra organizado en 3 partes: Teoría, Diseño y simulación de aplicaciones y Laboratorio Experimental. Al final se dispone de un apartado con prácticas con hardware, basadas en una sencilla y económica tarjeta en la que se desarrollan algunos proyectos de forma real.



Autores:

José Mº. Angulo Usategui: Dr. Ingeniero Industrial. Catedrático de Arquitectura de Computadores. Universidad de Deusto.

Aritza Etxebarría Ruiz: Licenciado en Informática. Jefe del departamento de informática. Colegio Vizcaya.

Ignacio Angulo Martínez: Licenciado en informática. Profesor del Departamento de Arquitectura de Computadores. Universidad de Deusto.

Iván Trueba Parra: Ingeniero en Automática y Electrónica Industrial e Ingeniero en Organización Industrial. Profesor del Departamento de Arquitectura de Computadores. Universidad de Deusto

El nombre del ganador será publicado en nuestra web y notificado por mail. Dispondrá de 30 dias corridos para reclamar su premio. Pasado este tiempo, se procedera a efectuar un nuevo sorteo en el numero siguiente de uControl.

µCONTROL
Electrónica en General Pics en Particular

Diseño y Diagramación
azimut.estudio@gmail.com / la plata / bs as / argentina